Multi-Party Electronic Payments for Mobile Communications

Michael Peirce

A thesis submitted for the degree of
Doctor of Philosophy in Computer Science
University of Dublin, Trinity College
Department of Computer Science
October 31st, 2000

Declaration

I hereby declare that:

(a) This thesis has not been submitted as an exercise for a degree at this or any other University.

(b) This thesis is entirely the work of the author, except where otherwise stated.

(c) The Trinity College Library may lend and copy this thesis upon request.

_____

Michael Peirce
October 2000

To my parents, for their support, love, and encouragement
throughout all my years of education

and

In memory of my Grandpa
Austin
who passed away as the final
chapters took shape.

Acknowledgements

First and foremost I wish to express my sincere thanks to my supervisor, Dr. Donal O'Mahony, Head of the Networks and Telecommunications Research Group (NTRG) here in Trinity College Dublin. Throughout my years of postgraduate research Donal has been a constant source of well-needed guidance, provocative discussions, and helpful advice. He always took the time, despite a busy schedule, to analyse my ideas and never failed to produce numerous suggestions for improvement. I must thank him for his patient reading of all the various versions of my write-ups and the resulting comments which have helped shape this dissertation. In addition, I am most grateful for the funding he arranged.

I wish to thank the current and previous members of the NTRG who offered their valuable opinions and knowledge throughout my time with the group. In particular I must thank Hitesh Tewari for all his advice over the years and for scrutinising the final manuscript. Hitesh had the misfortune of having an office next door to mine, and got bombarded with all my initial ideas. Some of those schemes that survived the ensuing hours of discussion, debate, and whiteboard scribbling, make up part of this thesis. I had the honour of co-authoring a book, entitled *Electronic Payment Systems*, with both Donal and Hitesh, and it provides material related to the background of this thesis. My sincere gratitude also goes to Dr. Linda Doyle, not only for her enlightening ideas, but also for designing a voice telephony application that was used in demonstrating this work.

I am also grateful to Professor John Byrne, Head of the Department of Computer Science in Trinity College Dublin, for arranging funding and providing me with the opportunity to share my enthusiasm for electronic payments by lecturing a number of postgraduate courses. The efficient support and advice provided by the staff of the Department of Computer Science has been invaluable. I am also thankful of the travel sponsorship provided by the Trinity Trust Foundation which helped me to attend the European Wireless Conference and the World Wide Web Conference to present papers relating to the thesis.

Many people, including anonymous referees, provided feedback on conference and journal papers related to this dissertation. I am grateful for their comments which have lead to a number of extensions and improvements on the final work. A number of invited presentations, and demonstrations of the prototype, were made to companies and research laboratories including Hewlett-Packard Labs Bristol, Hewlett-Packard Ireland, Eircom (formerly Telecom Eireann), Telenor, and ICL. Each yielded interesting discussion and useful feedback. Thanks also to NTL (formerly Cablelink) for airing a demonstration of the project on the local Dublin television channel.

Many thanks to my college classmates Kevin Bushe, Maurice Leyden, and PJ McKenna for reading this dissertation and providing valuable feedback.

Finally I must thank my family for their never-ending support and encouragement, and my friends for knowing when it was time to take a break and go out to play.

Abstract

Multi-Party Electronic Payments for Mobile Communications
Michael Peirce
Supervisor: Dr. Donal O'Mahony

As mobile communications become increasingly sophisticated and ubiquitous, traditional mobile billing with its implicit trust relationships will no longer be adequate. With a large number of different sized mobile networks, a huge variety of value added service providers and many millions of roaming users, it is desirable to remove any unnecessary trust in order to increase security and provide incontestable charging.

Billing allows each party involved in a call to eventually receive a share of the revenue generated. An examination of network billing techniques reveals a number of critical shortcomings and emerging problems. We address these issues by designing a multi-party electronic payment scheme that allows all parties involved in a call to be paid in real-time. The mobile user releases an ongoing stream of low-valued micropayment tokens into the network in exchange for the requested services. Dynamic pricing is supported by the association of a pricing contract with the call which specifies the cost of each leg of the call route. Any user with a mobile device and monetary value can use and pay for network access and services in any mobile network into which they roam. We eliminate the need to authenticate the user or contact a distant home network for billing purposes.

Extensions to the basic scheme provide mobile wallet functionality and allow user-to-user payments. In addition solutions are designed to cope with frequent handovers between independent picocells, to aggregate several payment streams into one in the core network, and to inter-work with networks using legacy billing techniques.

A detailed survey of micropayment techniques forms part of the design process and a number of new micropayment contributions result from observations made. In order to chose appropriate techniques for payment a performance comparison of micropayment schemes is presented based on benchmark measurements taken for the underlying cryptographic algorithms. The multi-party micropayment protocol is prototyped in a wireless environment where it is used to pay for user traffic from existing Internet applications.

The proposed scheme has the potential to revolutionise mobile communications by allowing the emergence of many independent inexpensive high-speed picocell networks and by allowing any network entity to sell value added services. Mobile users will be able to select the most appropriate access network and services wherever they roam, paying all parties for resources as they are provided.

Related Publications

Journal Papers:

M. Peirce and D. O'Mahony. Flexible Real-time Payment Methods for Mobile Communications. *IEEE Personal Communications*, 6(6):44-55, December 1999.

M. Peirce and D. O'Mahony. Scalable, Secure Cash Payment for WWW Resources with the PayMe Protocol Set. *World Wide Web Journal*, 1(1):587-602, O'Reilly & Associates, December 1995.

Books:

D. O'Mahony, M. Peirce, and H. Tewari. *Electronic Payment Systems*, Artech House, Boston/London, 1997.
Artech House Best Seller
Amazon.com Best Seller

Refereed Conferences:

M. Peirce and D. O'Mahony. Micropayments for Mobile Networks. In *Proceedings of European Wireless '99*, pp. 199-204, Munich, Germany, October 1999.
Awarded Best Paper of conference.

Technical Reports:

M. Peirce and D. O'Mahony. Multi-Party Payments for Mobile Services. Technical Report TCD-CS-1999-48, Dept. of Computer Science, Trinity College Dublin, Ireland, September 1999.

Contents

List of Figures

List of Tables

1 Introduction

"Discovery consists of seeing what everybody has seen, and thinking what nobody has thought."
Albert Szent-Gyorgyi.

1.1 Mobile Communications

The last decade has seen a dramatic increase in the availability and widespread use of mobile phones as a means of placing voice calls over a wireless radio link into the public switched telephone network. The number of mobile communications users continues to grow and is expected to reach one billion by 2004. Mobile devices are also increasingly being used for data services other than voice and as a means of accessing the Internet. It is predicted that eventually most devices connected to the Internet will be mobile computers [Per00].

Mobile communications are currently provisioned by a small number of large independent mobile networks with extensive coverage areas, often encapsulating entire regions or countries. Mobile users chose a single mobile network operator (NO) who provides connectivity and charges for usage. In order to allow their users to roam and use the services of a foreign mobile network, the home operator must establish a bilateral roaming agreement, for billing purposes, with that foreign network.

User payment for calls is always settled with the home network operator, including services used while roaming in distant networks. Usage bills are produced in the case of credit-based subscribers, while the accounts of prepaid users are automatically debited. The user charge for a call placed in a foreign network is set by the home operator. This can result in two users being charged significantly different amounts for making an identical call, based on which home network they come from. As part of the thesis we propose a solution which removes the need for payment to be handled by a single home operator and allows all users to be charged equal amounts within the same mobile network.

1.1.1 The Changing Environment of Mobile Communications

The next decade will see the emergence of ubiquitous mobile communications where every device will be capable of communicating over a wireless link. Mobile users will have constant connectivity through a number of mobile access networks using a variety of mobile communications protocols. There will be a large number of independent public and private mobile network operators, perhaps many thousands within a single city. The size of the different access networks will range from wireless in-building networks, to local area wireless networks for pedestrians, to wide area city and suburban cellular networks, to global satellite broadcast networks, as depicted in Figure 1-1.

In this environment users will always be in range of one or more mobile networks and will be able to select one that best meets their requirements at the time. Roaming between independent networks will occur daily,

even for those mobile users who never venture out of their home city. The mobile infrastructure for such a mobile communications environment will grow from the evolution of existing wide-area cellular communications together with the emergence of low-cost local area wireless technologies.

Mobile users will continue to pay for shared use of the scarce bandwidth provided by wide-area cellular communications. However, they will now also have the option of obtaining temporary access to the fixed network through a nearby local area wireless network into which they have roamed. A local wireless link can provide higher bandwidth and better quality-of-service than a wide-area mobile protocol and at a lower cost. The user benefits by obtaining a better less expensive service while the local provider gains by receiving revenue for providing wireless access. In turn, use of local independent picocells will alleviate some of the traffic load from wide-area mobile providers, allowing increased reliability and service for those parties who are not in range, or who are moving too fast to use a local wireless network. The emergence and success of local area providers will require that they can receive payment for the services they provide. Secure payment of entities within a mobile environment with many independent operators is one of the main concerns of this thesis.

Local area wireless providers will emerge from the enhancement of existing local area networks (LANs) with wireless capabilities and the widespread availability of low-cost wireless receivers. Companies and institutions will add wireless connectivity to their LAN, allowing their users to be constantly connected while moving within range of the premises or campus. The ACTS project COBUCO [COBU98, COBU96], with which we were involved, designed one such cordless business communications system based on the Digital Enhanced Cordless Telecommunications (DECT) [ETSI96c] standard to provide mobile connectivity in the workplace. We envision the unused excess bandwidth provided in these mobile LANs being provided on demand to passing mobile users, in exchange for payment. Unlike traditional LANs, mobile connectivity can be provided to parties outside the premises.

Local mobile networks will also become pervasive in public places such as shopping centres, restaurants, airports, supermarkets, and on busy streets, where they will provide mobile access to the public with appropriate usage charges. With the advent of inexpensive wireless technologies every party who can receive wireless traffic and relay it into the fixed network can potentially be a small picocell provider. High speed fixed line technologies [Sam00] such as cable modems and xDSL will provide every home and office with a permanent connection to the Internet. Wireless access to that fixed connection could be provided using any one of the low-cost wireless technologies such as Bluetooth [Haa00, Blu99, Haa98], HomeRF [NSL00], DECT [ETSI96c], wireless LAN technology including 802.11 [IEEE99] and HiperLAN [ETSI96d], and Infrared [Wil00].

Already there are a variety of commodity mobile communications devices including cellular phones, pagers, handheld computers, Personal Digital Assistants (PDAs), and laptops. Future mobile devices will have the ability to use several popular wireless protocols, and therefore will be able to connect to whichever network is most appropriate. For example, a mobile user might connect through a nearby stationary computer using Bluetooth, making calls and paying for them. The Cahners In-Stat Group forecasts that over one billion Bluetooth-enabled wireless devices will be produced by 2005. In the future, rather than implement each wireless protocol in hardware, it will be possible to use a software radio [CPP99, MBLT+99], where the protocols are implemented in software and can be downloaded and used as required. The availability of network independent mobile devices will allow roaming between mobile networks employing different communications protocols and will act as a catalyst for the emergence of ubiquitous mobile communications.

While there will be a large number of different network operators, both mobile and fixed, there will be an even greater number of independent value added service providers (VASPs). These will provide additional services other than transport of user traffic. For example a VASP might provide information services such as weather forecasts, street maps, stock quotes, and news. Services such as voicemail, online banking, and market trading will also be provided. Some VASPs will allow the purchase of material goods such as concert tickets, drinks from a local vending machine, or payment for a car wash. Indeed VASPs will ultimately provide any service that can be paid for. Like network operators, these VASPs will want to dynamically set charges on a per-call basis, and will not want to be constrained by NO pricing models. It should be possible for any entity with a network connection to provide services to users willing to pay for them.

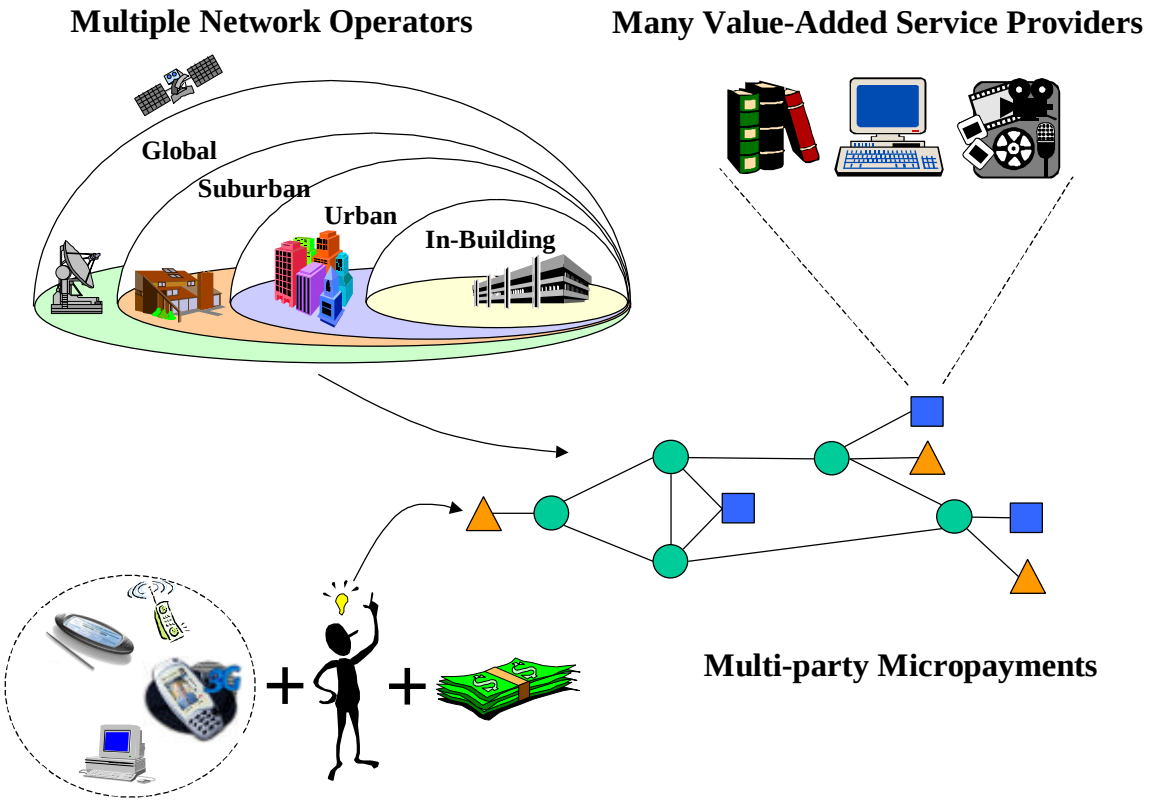


Figure 1-1 Envisioned Mobile Communications Environment with Real-Time Payment

Support of this envisioned architecture of future mobile networks can be found in the literature. The next generation of wide-area mobile networks, the 3rd Generation (3G) mobile system, allows for different radio access networks based on user mobility speeds [FHJ98, UMTS98]. Seamless roaming between different wireless network technologies is envisioned in [FRGH+99]. Ubiquitous connectivity through different sized networks is discussed in [CW99], where it is noted that current deployment is hampered by a lack of roaming agreements between independent networks. [PW00] puts forward the idea that the traditional payphone could evolve into a high-speed picocell basestation. The Ph.D thesis of Balakrishnan examines how to provide reliable data transport over heterogeneous wireless networks [Bal98b]. Finally the ICEBERG [WRCB+00] project is implementing a core network architecture for supporting diverse wireless access networks.

However a crucial consideration, not addressed by the research, is how to settle payments in a ubiquitous mobile communications scenario. As detailed in Chapter 2, current billing relies on the trust relationships established through roaming agreements. Operators must be trusted to generate accurate usage bills, and users must be trusted to later pay their home operator for services used while roaming. However with so many independent mobile networks of varying sizes it will be impossible for every network operator to

establish agreements with even a small fraction of other networks. Forcing each mobile user to have a single home network operator through whom all payments must be settled will therefore not allow freedom of roaming. Nevertheless, it must be possible for a mobile user to roam into any network and use its services.

The issue of operator trust is also important because the visited operator generates a record of the call, including its duration and services used, for payment purposes. However we have described how in a ubiquitous mobile setting any party can provide and charge for wireless access and services. Becoming a picocell operator will require as little as a Bluetooth network card and fixed network access. It will be impossible to trust all network operators, or to locate those that cheat. With such a large number of mobile operators, and a huge number of roaming users, billing based on trust will no longer be adequate or scalable.

The main goal of this thesis is to address the lack of security, scalability, and flexibility of current mobile billing systems. Our approach is to allow all parties involved in a call to be paid as they provide the services. The solution can be applied to existing mobile communications to eliminate user fraud, to provide accurate and dynamic charging, and to allow efficient flexible roaming in distant networks without need for roaming agreements or network connections to the home operator. While aiding the current mobile environment, such a solution will be a necessity in the envisioned future mobile scenario in order to allow roaming between all networks with fair charging and guaranteed payment.

1.2 Electronic Payment

The remarkable recent growth of the Internet has brought with it the need to perform commercial transactions over the network, thereby enabling electronic commerce. A key requirement of electronic commerce transactions is a method to allow payment to be made for any purchased item. When such a payment is effected electronically, by exchanging monetary value across a computer network, it becomes an *electronic payment* or *network payment*. Many electronic payment schemes have been proposed [OPT97], some of which are based on existing payment instruments such as credit cards and cheques while others create new forms of electronic value. However electronic payment research has largely been concerned with the problem of making payments to a single vendor across the Internet. In contrast, in this thesis we investigate whether electronic payment techniques can be applied to the very different scenario of mobile communications.

Telecommunications billing is based on the idea of creating a record of resource usage, and is usually performed by the entity who provided the resource and is due payment. In essence, billing is one party's unverified record of events. There is no proof that the transaction took place, that the stated amount of resources were consumed, or that the user was actually present. In contrast, electronic payments can remove the need to trust the biller and provide non-deniable proof of the transaction. Real-time payment is made by the user for a specified amount which cannot be altered by the payee. If feasible in the mobile communications environment, electronic payments may offer a better solution than traditional telecommunications billing, which has been based on the same principles for over one hundred years.

A typical mobile call will not be limited to the local access network but will connect through one or more fixed network operators and may use the services of VASPs located in any part of the network. In addition, other mobile networks may be involved for calls made to other mobile users. The vision of this thesis is that electronic payments can be used to pay all of the parties involved in providing services to a mobile user. If a user holds a mobile communications device and has electronic monetary value available through that device, he should be able to use and pay for network access and services in any mobile network into which he roams, as depicted in Figure 1-1. To facilitate this, we propose the first multi-party electronic payment system, which we tailor for the mobile communications environment.

1.3 Multi-Party Payment Scenario

With mobile communications a user might arrive in a new network, place calls which are routed through several independent networks, and use the services of both local and remote value-added service providers.

A typical scenario in which multi-party payments are required is now considered. A mobile visitor might drive into a new city. As he arrives at the city outskirts he places a call through one of the wide-area mobile networks to obtain a city traffic report and directions to his hotel. Later, having checked into his room, he uses the hotel local area wireless network to call an acquaintance, another mobile reachable through the city wide-area network, to inform her of his arrival. They make arrangements to meet in a cafe in a nearby shopping centre. Arriving early at the café, our visitor uses the local pedestrian network covering the shopping centre to place a long distance call, which is routed through two independent networks, to a remote VASP which provides him with voicemail services.

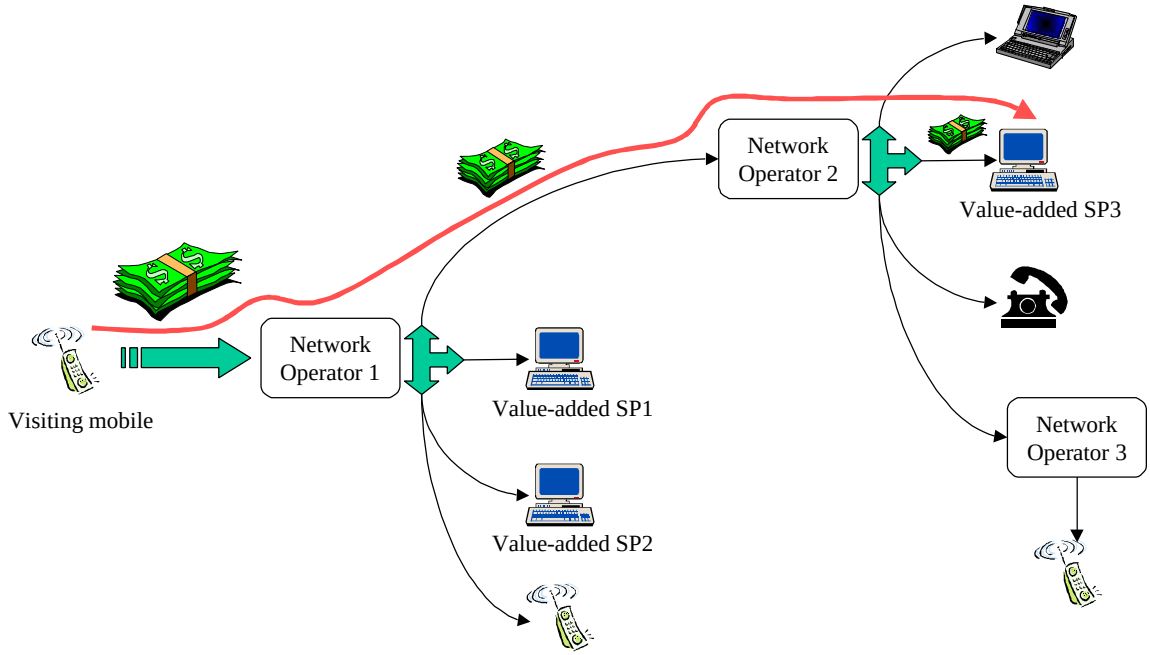


Figure 1-2 Multi-Party Electronic Payment in a Mobile Environment

In each call *multiple parties*, the NOs and VASPs, are involved in providing the call services. Figure 1-2 depicts the multi-party payment concept, where the mobile user pays all entities involved in a call as they provide the services. The diagram illustrates the final call placed in our example where two NOs and one voicemail VASP are paid. Payment will be ongoing as services are used, because the total usage is not known in advance, nor can a roaming mobile user be trusted to pay the full amount afterwards. Tariffs are set when the call is setup and are presented to the user device for agreement before the call begins.

The mobile releases a stream of prepaid tokens into the network in exchange for service. Each party providing service takes their share of the payment. This is done is such a way that every party gets paid the price they quoted at call setup, and no entity can steal another's value. Depending on the call requirements, payment might be made every few seconds, or for every kilobyte sent, or based on a combination of both elapsed time and traffic volume. Therefore the scheme will need to be able to handle repeated payments of small amounts, known as *micropayments*.

When mobiles pay in real-time the need to settle payment through a distant home operator is eliminated. By allowing each entity to be paid directly by the user, the need for numerous inter-connect agreements between

operators can be removed. In turn, this will allow the emergence of many independent VASPs as well as local area wireless operators, and ultimately ubiquitous mobile communications.

1.4 Scope of the Research

The focus of the research is to study the use of electronic payments in a mobile communications environment. The main resulting contribution is a multi-party micropayment scheme allowing a roaming mobile user to pay every party involved in providing mobile communications as services are used.

1.4.1 Outline of the Dissertation

The thesis is structured as follows. In Chapter 2 accounting and billing systems for both fixed and mobile telecommunications networks are examined. The way in which billing is being extended for next generation networks and for Internet traffic from mobile users are surveyed. The proposed different billing usage records for a wide range of scenarios including both network transport and application billing are also presented. The purpose of this chapter is to highlight the inefficiencies and inadequacies of billing. The problems need to be clearly understood and scrutinised before effective alternative solutions can be designed as part of the thesis.

The thesis proposes to use electronic payments to pay for mobile services, rather than using billing. Chapter 3 gives a thorough overview of electronic payment research. In particular the underlying cryptographic techniques from which the payment protocols are constructed are examined. It is shown that electronic payments aimed to provide high-value transactions, called *macropayments*, are too inefficient for our envisioned scenario. A detailed review of the micropayment literature is then performed, in order to understand all the known techniques available to allow efficient repeated payments. A taxonomy of micropayment schemes, based on the cryptographic constructs used, is given. To the best of our knowledge no other detailed micropayment survey and analysis has been previously published. Based on observations made during the survey we make three new contributions to micropayment state-of-the-art.

While Chapter 3 presented the theory of micropayments, Chapter 4 aims to compare the performance of the different schemes. Benchmarks are presented for the basic cryptographic constructs upon which all electronic payments systems are based. Knowing the constructs used in each scheme from Chapter 3 it is then possible to calculate the performance of each micropayment scheme. This study allows the estimated performance of each scheme to be compared. From this it can be seen how certain micropayment techniques perform, and this knowledge is later used in the design of our own micropayment scheme. It is also shown how poorly macropayment schemes perform, when compared to micropayments.

In Chapter 5 we present the first multi-party micropayment protocol. We lay down requirements which eliminate the weaknesses of current billing protocols. The details of the protocol are presented and its operation in a typical multi-party mobile communications environment is explained. The security of the protocol is analysed and its performance is estimated and optimised.

While the majority of proposed micropayment protocols have not been implemented a demonstrator of our multi-party protocol was built, in order to validate it and gain experience. Chapter 6 describes the design and implementation of the multi-party micropayment demonstrator. Integration of a smart card component is discussed. The wireless scenarios in which the demonstrator was tested, and the way it was integrated with existing applications, are explained. Performance measurements are analysed, focusing on the computation and communications cost of the implemented protocol.

The basic multi-party micropayment protocol is extended to operate in a variety of different scenarios with increased flexibility in Chapter 7. Modifications are made to allow user-to-user payments, as well as efficient

payment for local services such as for items from a vending machine. A method of aggregating several payment streams together to form a single combined payment, allowing scalability in a heavily loaded core network is described. Another technique allowing uninterrupted payment during frequent handover between different independent networks, such as would happen when roaming between Bluetooth basestations, is designed.

Finally, in Chapter 8, the contributions of the thesis are summarised. A discussion of possible future work arising from the multi-party micropayment scheme is given. Initial results of this thesis have already been presented at the European Wireless ’99 conference [PO99b] and in the IEEE Personal Communications journal [PO99c].

2 Accounting and Billing for Network Services

"Immortal gods, I crave no pelf; I pray for no man but myself;
Grant I may never prove so fond, to trust man on his oath or bond."
From "Timon of Athens" by William Shakespeare.

2.1 Introduction

All network service providers, whether they are mobile or fixed operators or value-added providers, need to be remunerated for use of their resources. Recording resource usage for this purpose is called *accounting*. *Billing* is the process of applying *tariffs*, or set prices, to the usage details, in order to generate an invoice for payment.

Two central problems with telecommunications billing are the cost of the actual billing process and the high level of fraud. Billing costs arise from the need for dedicated metering equipment and networks for transferring accounting records, billing software, business planning and fraud analysis. This alone is estimated to cost the industry over €7.6 billion a year by 2004. In addition to this are the recurring individual costs per bill for handling, paper, postage, payment processing and clearing, estimated at between €5.00 to €10.00 per transaction [Eng98]. Taking that there will be over 5.3 billion phone bills produced in the U.S. in 2005, improvements are clearly needed. While Electronic Bill Presentment and Payment (EBPP) promises to reduce costs, another option used in our approach is to allow prepaid services. Call usage can be maintained by the user device if a bill-like summary is still desired.

A user must trust the accuracy of usage records and their bill. Post-fact billing and complicated tariffs make it difficult for a user to recall their precise usage, let alone dispute it. Even still, OFTEL, the UK telecommunication regulator, received over 25,000 complaints about inaccurate bills in 1999 [OFTL00], an increase of 80% over the previous year. We believe this problem will escalate further, especially as the number of independent service providers increase.

The U.S. Federal Trade Commission has highlighted the widespread problem of fraudulent billing [FTC98]. However, this is only one of many aspects of telecommunications fraud [Col99a, Col99b, Col00, ES97], which costs the industry an estimated €12 billion annually. The average telecommunications company loses 3 to 5% of their total revenue to fraud [Sey98, ES97]. Cellular network fraud has become a major problem, and efforts to curb the extent of this fraud have led to imposition of service restrictions on users, particularly in regard to roaming. While work has been done to detect fraudulent call patterns [Col99b, HE98], the core problem is the trust needed for usage billing and the unlimited credit it provides.

In the remainder of this chapter the accounting and billing techniques used in both mobile and fixed networks are examined. We aim to show that basically all systems use the same technique of creating a log of resource usage. The only real difference between the different billing formats is the contents of that log record. In Section 2.2 we describe how billing originated with fixed telecommunications, and Section 2.3 shows how it was extended to mobile networks. We highlight the inefficiencies and problems of this approach, including the way it has been applied for prepaid services. Section 2.4 describes how new record formats have been defined for mobile data usage. The approach taken in next generation mobile networks uses the same approach but does propose some fraud reduction techniques, as described in Section 2.5. As payment for Internet usage emerges, the same technique is being applied in that domain. The initiatives for Internet telephony, roaming Internet users, and general IP traffic and applications are presented in Sections 2.6, 2.7, and 2.8 respectively. Finally we conclude in Section 2.9 by showing how these new proposals will work together, and explain the problems that remain unaddressed.

2.2 Fixed Telecommunications Billing

The practice of recording the details of individual calls for billing purposes has been in use since the commercial deployment of early manual telephone exchanges in the late nineteenth century. In those times *Call Detail Record (CDR)* information for long distance calls was collected and recorded by cordboard operators at the exchange [Flo97, Col99a]. The operator manually wrote the details onto a specially formatted record called a *toll ticket*. These tickets were later sent to a clearing office where customer bills were generated.

The same basic principles for charging for the use of telecommunications networks are in place today. The telephone exchanges have evolved into complex digital switching systems totally controlled by software. When a subscriber places a call, the local exchange automatically creates a record of the call details [RV94], a process known as *automatic message accounting (AMA)* or *toll ticketing (TT)*. The CDRs are stored in a file at the local exchange and periodically sent to a centralised billing system, usually at another location. This off-line procedure can vary from physically transporting magnetic tapes to transmitting the records across a data network. If the records are transferred immediately, the process is known as *hot billing* or *on-line charging*. This allows bills to be processed on request or within a given time limit. However, it is not yet in widespread use among operators.

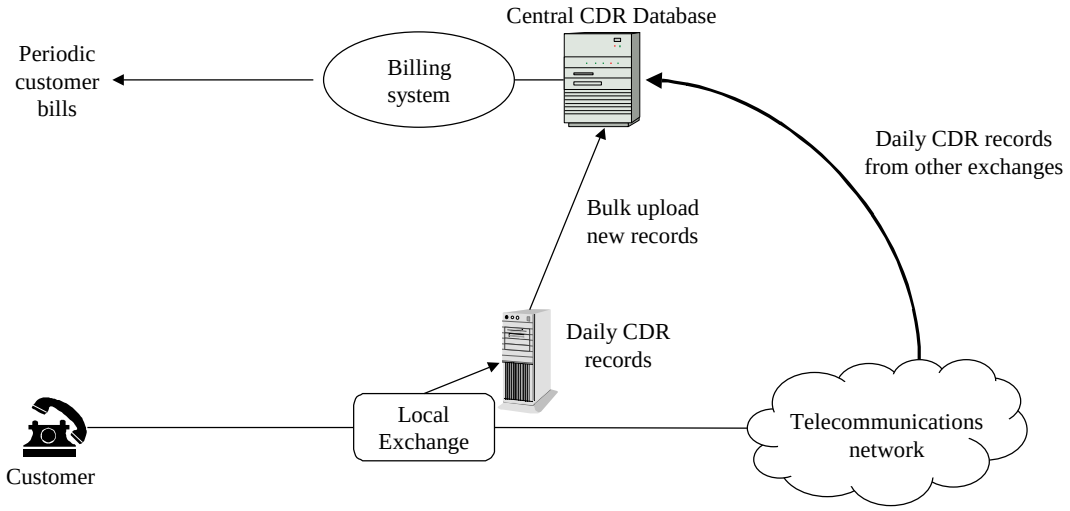


Figure 2-1 Billing Cycle in Existing Telecommunications Networks

Billing software extracts information from the CDRs and calculates the cost of the call for the customer by applying tariff tables based on called distance, call duration, time of day the call was placed, and subscriber

type (residential or business). Rates may also vary according to subscriber charging plan, history of previous calls, and promotional discounts. Every billing period a bill, possibly detailing the calls placed, is sent to the customer for payment. The billing lifecycle is summarised in Figure 2-1.

The contents of a CDR vary from operator to operator. This is partly due to their use not only for billing but also for the measurement of traffic and quality of service. Switch architectures from different manufacturers also produce information in different formats. Recent standards for call records [ETSI99c, Telc99, ETSI98c, ITU98a] do exist, but are not yet widely adhered to. However, there are some critical fields for billing, usually present in some form. These are:

- **Source of call**. The CDR must uniquely identify the party to charge. Normally, calls are billed to the owner of the phone from which a call is placed, identified by the calling line number. Other scenarios might bill the called party number (reverse charging), an account number (calling card or credit card services), or a personal user identity (allowing personal mobility).

- **Destination of call**. Usually the called number (digits dialled) and a translated number if appropriate. Translated numbers arise from call forwarding and special rate numbers (toll free, premium rate, local national rate). The distance rate to apply for billing is obtained from this information.

- **Time and duration**. The date and time the call took place and the length of time it lasted.

- **Routing points**. The identity of resources and equipment, such as trunk lines or gateways, used during the call. Distance billing may be partly based on the routes taken by the call, due to the interconnection agreements between different network operators for resource usage. Traffic analysis is also based on routing details.

- **Service information**. The type of call service used, such as basic call, premium rate, toll free, conference call, or other supplementary services. It may also include the quality of service (QoS) provided, such as use of a low delay optic fibre link instead of a slower satellite connection.

Charging and pricing information is usually applied when the records reach the billing system. The size of a CDR can range from 20 bytes to several hundred bytes, but for performance efficiency it should be kept to a minimum. If a call is not answered, a CDR may still be created, although the customer will not be billed for these. Sometimes two or more records are created for the same call, an originating record and a terminating or trunk record. This is useful when all the necessary information is not available at one place in the network.

CDRs are being standardised as the method for charging and billing in both fixed and mobile networks. New record formats and contents must be defined for each network type or service, as illustrated in the following sections. The European Telecommunications Standards Institute (ETSI) have outlined the use of CDRs for PSTN, ISDN, Intelligent Networks (IN), cordless, mobile and packet networks [ETSI98a]. ETSI has also made CDR recommendations for basic Broadband-ISDN (B-ISDN)[ETSI96a], Universal Personal Telecommunications (UPT) [ETSI95] and European Telephony Numbering Space (ETNS) services [ETSI98b].

The only proof that a call took place is the CDR, and sophisticated duplication schemes are used to ensure that this data is not lost. The raw data generated by the switch is also kept for the purpose of settling disputes. However, there is no mechanism in place to prove the authenticity of the data. The call records can be denied by the customer or falsified by the operator.

2.3 Billing in Mobile Networks

Rapid growth in mobile communications has given rise to a large number of independent network operators, spanning many different geographic areas and countries. When these operators use a common mobile standard, it is possible to allow subscribers to roam from the home network to a visited location, choosing between the new operators available. Digital mobile networks [PGH95, Wal99a] with roaming capabilities include PDC, CDMA (IS-95) and US TDMA (ANSI-136), but over 70% of the world's mobile networks use the European standard GSM.

2.3.1 GSM

The Global System for Mobile Communications (GSM) [ETSI96b, RWO95, Wal99a], with over 425 network operators world-wide provides more than 332 million customers with the ability to make and receive calls when outside their home network. It is predicted that by 2001 one in twelve people in the world will use GSM. In order to co-ordinate the initial deployment of the GSM standard, fifteen mobile network operators signed a *memorandum of understanding (MoU)* committing to introduce GSM systems by 1991. This early agreement has evolved into an international association of GSM network operators, the GSM Association. Its purpose is to guide the commercial development of GSM, while working alongside the technical standards bodies such as the European Telecommunications Standards Institute (ETSI).

Billing and accounting procedures arising from international roaming are regulated by the GSM Association. Charges made to a roaming subscriber in a visited network must be collected by his home operator with whom he has a subscriber agreement. When the visited network operator provides services to the subscriber he needs to be assured that the subscriber's home network will compensate him for the charges incurred. A mobile user has no contract or relationship with the visited operator. For this reason, roaming is only permitted between networks that have arranged a *bilateral roaming agreement*. The GSM Association Billing Administration and Roaming Group (BARG) has laid down the regulations applicable to such roaming agreements.

The visited network operator provides services to roamers without the need for additional subscriber contracts or credit authorisations. To ensure that the subscriber is genuine, his home location register (HLR) is contacted. Authentication is performed based on the knowledge of a shared secret between the HLR and subscriber. Upon a valid identification, the subscriber can then make and receive as many calls as desired through the visited mobile network. CDRs are generated by the visited network and later forwarded in bulk to the home operator for settlement. The network operators settle payment for the resources used between themselves and the charge is ultimately reflected in the user's bill.

Multiple CDRs, or toll tickets, can be generated for different legs of a GSM call. These include the originating, terminating, and roaming call components. In a mobile-originated call the calling party usually pays for all stages. When a roaming mobile is called, the subscriber pays for the component of the call from the HLR to his location in the visited network. With optimal routing [CLR97], where calls are routed directly instead of via the home network, the roaming component can be minimised. Figure 2-2 shows a mobile subscriber A, in his home network, calling a roaming subscriber B. A will be billed for the originating leg (TT1), while B will be billed for the roaming leg (TT2) and terminating leg (TT3).

There are 17 different types of GSM toll tickets and the one for a mobile originated call contains 55 possible fields [ETSI99c]. The International Mobile Subscriber Identity (IMSI) identifies the source of a mobile originated call. In addition to regular CDR fields, the toll ticket contains location area, cell ID, radio channel allocation and international mobile equipment identity (IMEI).

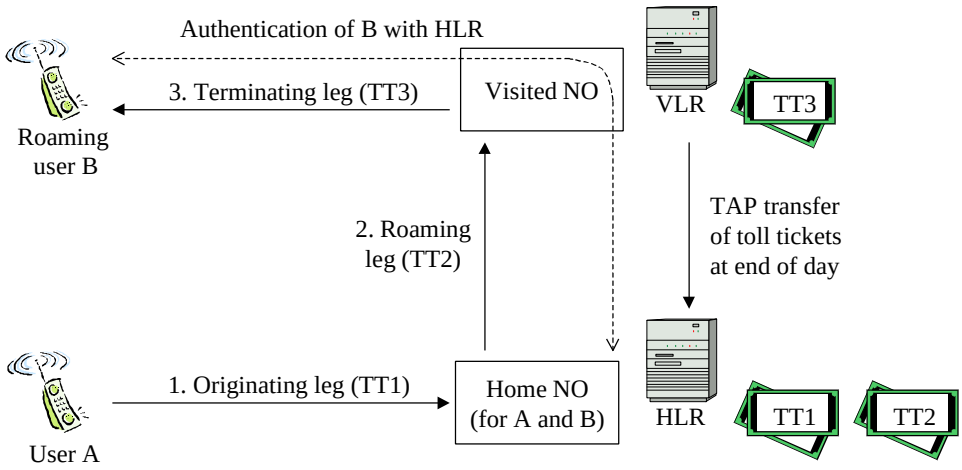


Figure 2-2 Generation and Exchange of Billing Records in the GSM System

The Transferred Account Procedure (TAP) [GSMA98] details how the CDRs of roamers should be transferred from the visited network back to the home network. The file formats and transfer methods between operators are specified by the GSM Transfer Account Data Interchange Group (TADIG). The latest version, TAP3 [GSMA00], caters for variable length records and allows individual erroneous CDRs to be rejected. Electronic Data Interchange (EDI) is used to exchange the TAP files in a standard message format. This is often performed using file transfer over an X.25 or TCP/IP network.

Mobile networks not based on GSM technology use procedures other than the GSM TAP to transfer the call records of roaming subscribers. The Cellular Intercarrier Billing Exchange Record (CIBER) [CINT97] is used for roamer billing by operators of CDMA, US TDMA and analogue AMPS networks. CDRs of roamers are converted into one of fourteen CIBER record types, depending on the call properties, before being exchanged. The CIBER standard is published by Cibernet, a provider of financial settlement services for wireless operators.

The roaming agreement is used to establish trust between independent operators. However there is no guarantee of incontestable charging or payment for any of the parties involved. In addition, each network operator may have to exchange CDRs and payment with up to several hundred other network operators. In 1999 there were over 25,000 roaming agreements between GSM operators. Such direct reconciliation is both costly and inefficient. In order to minimise the number of transactions, and hence the cost of settlement, a central *clearinghouse* or broker can be used. Even with hot billing, fraudulent calls may last for hours or even days, before they are detected when the toll ticket is finally cleared. To reduce such fraud, some operators will terminate all calls that exceed a specific time limit. Other problems such as privacy issues also exist. Use of CDRs result in databases containing details of the location and duration of every call a user makes or receives.

2.3.2 Retail and Wholesale Rates

The fixed network is based on a dual price system. For each call, the user is charged a retail price, called the *collection rate*, by the originating network operator. In turn, the originating NO is charged a reduced wholesale price by the terminating operator for completing the call connection. For a domestic interconnection between two operators within the same country, this wholesale price is known as a *termination charge* and is based on the cost of delivering the call to the final destination within the termination network. Thus the charge may differ depending on what part of the network the call is going to or what resources are used.

The current situation with international interconnection works differently. The originating and terminating international operators agree on a wholesale price, called the *accounting rate* [ITU98b], for delivering traffic over their part of the international link. Traffic usage of the link is recorded and if there is an imbalance in the volume of incoming and outgoing traffic then the originating operator which generates more traffic pays for the difference, called the *net settlement payment*, to compensate the terminating operator. The *settlement rate* is usually half of the accounting rate, which assumes that the cost of terminating the call is the same for each partner.

The accounting rate system has come under criticism [Ber98] and is currently being reformed. It was originally designed for an industry structure based on national monopoly providers and is biased against countries which send more calls than they receive. International calls to mobile networks introduce further problems. The settlement rate is set on the basis of fixed network termination costs. If the international call must be routed into a mobile network, with an additional domestic termination charge, the settlement rate may not be high enough to cover this charge as well as the costs incurred by use of the international facilities. Hence one or more of the operators involved could conceivably lose money.

For both domestic and international interconnection, the seconds of traffic terminated for each call are recorded in CDRs. These are added together at the end of the billing period and charged at the wholesale price to the originating operator. The GSM Association have introduced a wholesale tariff between GSM operators for roaming services, called the Inter-Operator Tariff (IOT). Two roaming users, each from different home networks, who make an identical call in the same network, will be charged two completely different amounts. While this can be used to avoid complicated roaming tariffs, it seems biased against the users. In Chapter 5 we argue that it is fairer to allow all roamers to pay equal amounts, by paying in real-time.

2.3.3 Pre-paid Solutions

To increase the probability of receiving payment, the above billing schemes all rely on a strong legally binding contract being established with the user. Where such a contract is not desirable, pre-paid solutions allow the user to avail themselves of specific services, which they pay for in advance. Calls are cut-off in near real-time, when the pre-paid amount has been used up, thus preventing a possibly unexpectedly large bill accumulating.

The coin-operated payphone was one of the first pre-paid solutions for the telephone network. Problems due to theft of the deposited money, the cost of collecting the coins and counterfeit fraud led to the development of card operated payphones. Pre-paid cards, usually based on memory cards or smart card technology, are purchased from a distributor. As the card is used, its value is decremented locally, often in response to toll pulses sent from the exchange. There is no need to verify the card online with a central database, as is the case when credit cards are used in payment.

Calling cards offer a temporary account with a network operator against which calls can be made. Such accounts can be pre-paid or credit based. In the former case, the account status needs to be monitored and the call terminated in real-time when the value is used up. International discount calling services are similarly account based. They allow the use of an alternative network, and its reduced tariffs, by connecting to a user and presenting them with call-out services from that alternative fixed network.

Prepayment becomes more complicated in mobile networks because it is still desirable to be able to receive calls and roam internationally using many different network operators. The majority of pre-paid mobile solutions are based on temporary accounts, maintained at the home location. Each solution usually has a

substantial initial charge to cover costs of registering and maintaining identities, making it infeasible for use by a casual visitor. Ericsson is one supplier of such solutions in Europe. Its range of GSM prepaid solutions are Intelligent Network (IN) based with hot billing to allow automatic call termination shortly after the account value reaches zero. Alternative GSM-based architectures for implementing the temporary account functions are described by Lin [LCR00], who notes their unsuitability for roaming.

Prepaid international roaming imposes additional difficulties because the prepaid account, maintained at a distant home network, must be decremented in real-time as calls are made. Initial solutions were ad hoc proprietary schemes. For example, some systems required a credit card against which to bill roaming charges; another used a flat-rate call back service through the home network; and another required that the GSM short message service (SMS) be used to send the desired number to the home network for call completion.

Recently, prepaid roaming solutions have been proposed based on the Customised Applications for Mobile network Enhanced Logic (CAMEL) [ETSI00g] features. CAMEL allows home NOs to provide operator-specific services to their roaming subscribers. When a roaming user initiates a call, the CAMEL process in their home network is contacted by the visited network for further instructions. The charging-related instructions allow the home network to set a limit on the call duration or volume; send data for inclusion in the visited network CDR; and request hot billing. While CAMEL can implement prepaid roaming, an online network connection is required to the home network at several points: before, during, and after every call. In Chapter 3, we discuss the scalability problems of online communication to third parties for payment purposes. The congestion problems of hot billing roaming users is also noted by Fang [FCL99], who uses queuing theory to group records before clearing.

2.4 Mobile Data Billing

While mobile networks were originally designed for voice, the ability to access the Internet or corporate Intranets over a wireless link is now required. By 2005 it is predicted that more mobiles will be connected to the Internet than PCs [KH00], and will be used by over 50% of the world's industrialised population. This will account for up to 30% of all Internet traffic. GSM data communications was initially provided by the Short-Message Service (SMS), Unstructured Supplementary Services Data (USSD), circuit switched data (CSD) over the voice channel, and the High-Speed Circuit-Switched Data Service (HSCSD) [Wal99a]. A faster shared packet switched service is now provided by GPRS, as described below.

2.4.1 WAP Events

There is a need for wireless protocols to address the limited bandwidth and small screen capabilities of current mobile devices. The Wireless Application Protocol (WAP) [WAP98, Peh00], was designed by an industry consortium, as one such solution. The WAP approach was to redesign the Internet protocol stack, replacing each layer with one or more lightweight layers designed for wireless applications. A WAP gateway is required at the NO to translate between the WAP protocols on the mobile side, and traditional Internet and Web protocols in the fixed network. A mobile user browses Web content from a WAP server or translated content from a regular HTML server, typically using a CSD bearer.

Billing for use of radio resources takes place as before using TAP, but the content must also be paid for. The WAP specification does not address this, although in late 1999 a WAP billing expert group was established. The approach is to generate WAP CDRs, including the URLs visited and volume sent, at the WAP gateway. These are exchanged and settled as with traditional GSM CDRs. WAP has been criticised [Kha99, Ban00b] for removing IP and re-inventing the entire stack. Alternatives exist, such as i-Mode from NTT DoCoMo in

Japan, Web clipping from Palm Computing, and LEAP [Ban00a] from the Free Protocols Foundation. However, they all also approach billing by recording every event during a session with CDRs.

2.4.2 GPRS Billing

The GSM General Packet Radio Service (GPRS) [GW99, ETSI00e, KMM00] provides wireless packet data access from a mobile node to external packet networks. It is used in both GSM and TDMA/IS-136 to provide wireless access to IP-based networks. Users share a dynamically allocated pool of packet data channels, one per GSM time slot, with a maximum throughput of 115 kbit/s. Since it is packet based, rather than circuit switched, the radio capacity is only used when actually transmitting or receiving data. Accordingly, new GPRS CDRs [ETSI00c] have been defined to record volume usage for charging and billing.

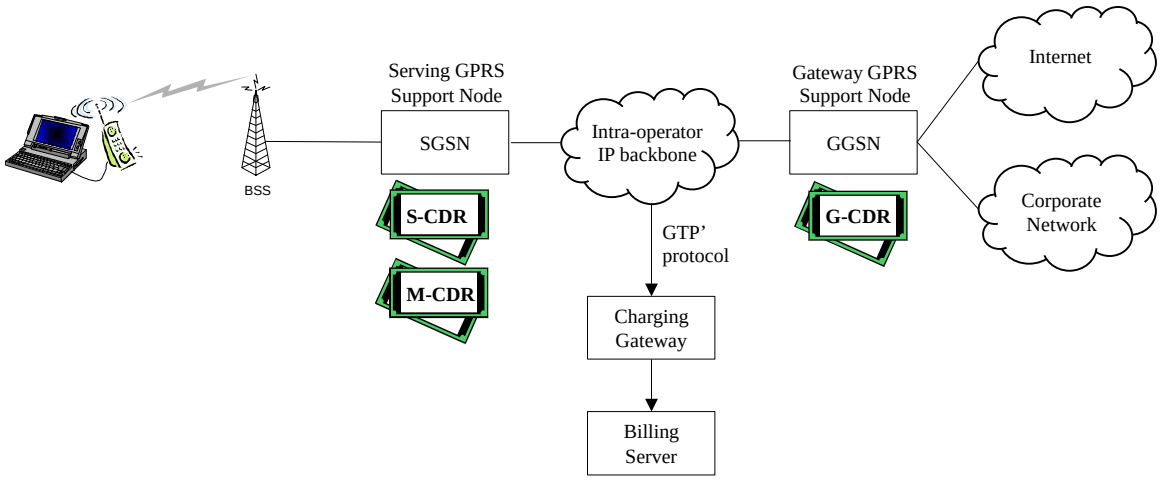


Figure 2-3 GPRS CDR Creation

Three different GPRS CDRs, generated at different network locations as shown in Figure 2-3, are required. The mobility management CDR (M-CDR), containing location and identity information, is generated when the user first switches on their GPRS terminal and attaches to the network. The SGSN CDR (S-CDR), and the GGSN CDR (G-CDR) are created at the start of a GPRS session. The S-CDR records the radio network usage including the QoS, while the G-CDR remembers the external IP network usage. Combined, the CDRs provide the total number of bytes sent in each direction, the radio capacity offered, the time and duration of the session, and the identity of the external network used. Time-stamped data is added to the CDRs throughout the session to reflect changes in the radio capacity or tariff time. If the IP destination address is optionally recorded, new CDRs are needed for each change of address. The GTP' charging protocol [ETSI00c], an extension of the GPRS Tunnel Protocol (GTP), is defined to transport the CDRs reliably, using UDP/TCP over IP to a charging gateway, where tariffs are applied.

The Enhanced Data Rates for GSM Evolution (EDGE) [FNO99, FMMO99] is a radio modulation technique allowing over 384 kbit/s for packet data. This is done by replacing the original GSM modulation technique of Gaussian Minimum Shift Keying (GMSK) with eight-phase-shift keying (8-PSK) modulation. The GPRS CDRs may be used with EDGE, although link adaption, with dynamic switching between coding and modulation schemes, will result in more frequent CDR updates.

Mobile data increases the number, size, and complexity of CDRs. The new records must remember every change during a session to allow accurate billing. In addition, no mechanism is provided for charging for use of the IP services and applications.

2.5 Future Mobile Systems

GSM and other existing digital cellular networks are second generation systems, the first generation being analogue based [Hil95]. The *third generation (3G)* system will integrate both mobile and fixed networks to allow personal and terminal mobility on a global scale. Multiple radio access networks will allow speeds of 384 kbit/s in the wide area, and up to 2 Mbit/s within buildings. In Europe, the name given to the future third-generation system is the *Universal Mobile Telecommunications System (UMTS)* [Oma98]. The ITU are standardising 3G systems under the name of International Mobile Telecommunications-2000 (IMT-2000). While 3G systems will have both PSTN and Internet components, we use the term "Fourth Generation (4G) system" to refer to an all IP-based converged network. Voice, data and network signalling is carried in IP packets both over the mobile networks [CGKT+00] and in the core network [PD00, 3GPP99b, 3GPT00].

2.5.1 3GPP Charging and Billing

The 3rd Generation Partnership Project (3GPP), an international consortium of telecommunications standards bodies, was formed, in December 1998, to produce technical specifications for UMTS. The 3GPP strategy is to build on existing GSM systems [CMO99] and they recommend the use of GSM/GPRS billing [ETSI00a, ETSI00f]. TAP3 records will be enhanced to cater for usage of new resources and services [3GPP99a]. However, variable bandwidth in a multimedia call, such as that caused by adding or removing video streams, will result in a large number of CDRs. Long calls must also generate several CDRs, to allow timely hot billing. In Chapter 5 we show that, by paying in real-time, such changes do not need to be recorded.

We note that the 3GPP has rejected the use of a Public Key Infrastructure (PKI) [Lan98] to provide security in UMTS [3GPP00, HMV99] after initial research proposals [USEC99, ASPT98]. In particular, shared user-home secrets are used to provide authentication, as in GSM, rather than user digital signatures. We also believe that user certificates are not suitable for real-time payment and discuss this further in Chapter 5.

The problem of allowing a roaming user unlimited credit is recognised by the 3GPP [ETSI00b]. The CDRs of roaming users take several days to reach the home location, which only then becomes aware of the amount of resources used. One proposal is that each visited network perform an online credit authorisation with the home network of every roaming user at the start of a call session. Essentially, this is the same as an online vendor-performed credit card authorisation, as described in Chapter 3, and suffers from the same scalability problems. The home network will limit the credit given to a user during the check. The visited network will then download a charging control algorithm, from the home network, to impose this limit. The charging algorithm calculates the amount the user has spent, in real-time, based on the tariffs from the home network. Applying the many different home network tariff policies in real-time will be a computational burden to the visited network. The program can be run by the visited network or by the user's own mobile device. Only the actual visited network will be able to prevent further resource usage when the spending limit is reached. The accuracy of the cost-control mechanisms will be approximate and will not reflect the precise real-time status of current charges. This online charging proposal has not been included as part of the UMTS specifications.

2.5.2 3GPP2

The CDMA mobile network community have formed the 3rd Generation Partnership Project 2 (3GPP2) to standardise 3G networks based on CDMA technology. 3GPP2 wireless data [MH00, PD00, 3GPT99] will be provided using Mobile IP [Per96], an IETF standard for handling wide-area mobility at the IP layer. Billing will be performed with the same IETF authentication, authorisation, and accounting (AAA) protocols, as used with roaming Internet users. These are described in Section 2.7. However, this architecture results in two layers of online authentication and billing, one performed with the home mobile network, and the second performed using AAA protocols with a home ISP provider, as depicted in Figure 2-9.

2.5.3 Billing with Non-repudiation

The security risks of allowing operators to generate CDRs has been highlighted. As depicted in Chapter 1, future mobile systems are likely to consist of a large number of separately administered access networks. An even larger number of competing *value-added service providers (VASPs)* will supply on-line content and services. The implicit trust relationships of existing billing mechanisms will need to be replaced. Initial attempts to provide *non-repudiation* of a bill, or undeniable proof of its correctness, are now outlined. Our own method of using real-time payment removes the need for non-repudiation of CDRs, by moving that need to the payment tokens.

The Advanced Security for Personal Communications Technologies (ASPECT) project [ASPT98] was a three-year ACTS project which investigated secure billing for UMTS applications, amongst other security features. ASPECT demonstrated an incontestable charging procedure, designed to allow small payments for value-added services [HP98, MPMH+98, HHMM+98].

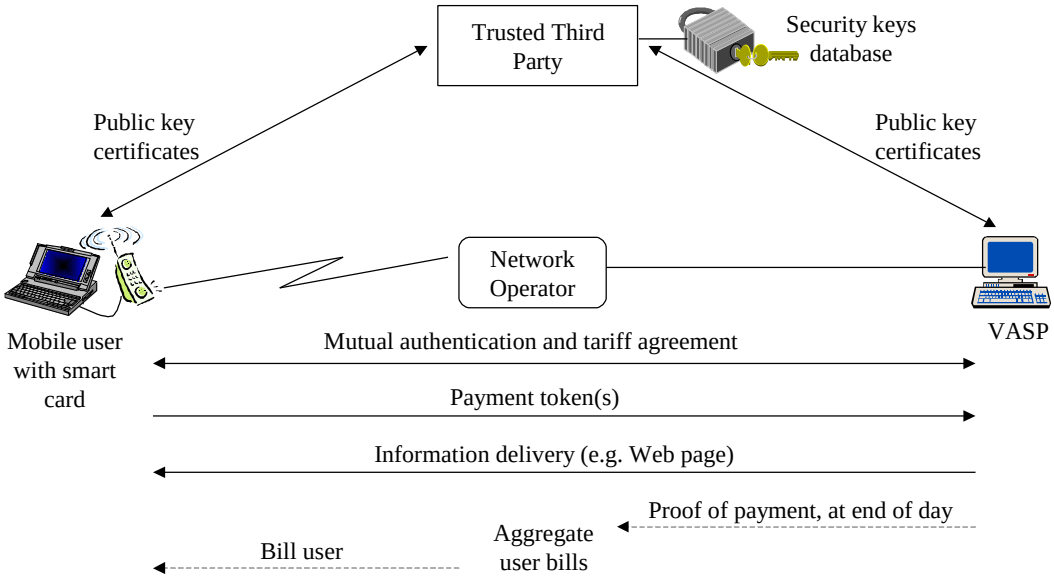


Figure 2-4 Secure Billing in ASPECT

The ASPECT approach was to break a call into two chargeable components, as also envisioned by the GSM Association [GA97]. The first component was the basic charge for bearer services, the transport of call data, provided by the network operator. This is handled using traditional billing. The second chargeable component was the premium rate charge for use of services provided by a VASP. The ASPECT solution allows the mobile user to make many small payments directly to the VASP, as the services are provided. The scheme is outlined in Figure 2-4. Each payment token can only be generated by the user and is proof that he agrees to pay the VASP a small fixed amount. The payment details are described in Chapter 3. At the end of the day, the VASP forwards the payment proof to the user's UMTS service provider, who then bills the user in the traditional fashion.

ASPECT improves on current solutions by providing incontestable charging for content services, guaranteeing that the bill from a VASP is genuine. However, it still does not address network operator billing, nor does it guarantee payment from the user. A scheme proposed by Zhou [ZL98], which does not consider VASPs, uses the same token method as ASPECT to guarantee correct billing from a visited NO. Essentially, the visited NO is acting as the VASP in this scenario. An alternative solution is proposed by Wang [WW98], where the user securely sends the bill back to the home NO, who in turn acknowledges its

receipt with the visited NO. However, this results in frequent online communication with the distant home network. The user may also use some service and never report the bill due.

2.6 Internet Telephony

Traditionally, voice calls have been carried entirely over circuit-switched networks, and the billing systems have been designed specifically for such a scenario. The popularity of the Internet has lead to voice-over-IP (VoIP), where all or part of a voice communication is carried over an Internet Protocol (IP) [DAR81, DH95, Com00] packet network. Polyzois [PPYS+99] considers the issues of migrating to Internet telephony, and suggests that the same general techniques used to bill roaming Internet users, described in Section 2.7, can be applied to billing for VoIP. Another approach, now examined, is to define new CDRs specifically for VoIP traffic.

2.6.1 Open Settlement Protocol (OSP)

The ETSI Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) [ETSI99b] project is addressing the convergence of voice over circuit switched and IP packet networks. The Open Settlement Protocol (OSP) [ETSI00d], proposed by TIPHON, defines how inter-domain usage, pricing, and authorisation information is exchanged between Internet telephony operators.

For usage information, the OSP defines a CDR format, referred to as a Usage Detail element, with fields specifically for IP telephony. There are fields to allow the billing unit to be in seconds, packets, or bytes, and several different address types, including e-mail, IP addresses, H.323 identifiers [ITU99a], or E.164 telephone numbers [ITU97b]. The OSP adds digital signatures to CDRs, preventing tampering during the clearing process. However, there is still no mechanism to prevent inaccurate CDRs being generated.

HTTP Header

```
POST osp/cdr HTTP/1.0
Content-type: multipart/signed;
    protocol="application/pkcs7-signature"; micalg=sha1; boundary=bar
Content-length: 1228

--bar
Content-type: text/plain
Content-length: 917
```

XML Content

```
<?xml version='1.0'?>
<Message messageId="123456789" random="87654321">
    <UsageIndication componentId="9876567890">
        <Timestamp> 2002-03-17T17:03:00Z </Timestamp> <Role> source </Role>
        <TransactionId> 6789098723 </TransactionId> <CallId> 1234432198766789 </CallId>
        <SourceInfo type="e164"> 81458811202 </SourceInfo>
        <DestinationInfo type="e164"> 4766841360 </DestinationInfo>
        <UsageDetail>
            <Service/> <Amount> 3 </Amount> <Increment> 60 </Increment> <Unit> s </Unit>
            <StartTime critical=”false”> 2002-03-17T17:03:00Z </StartTime>
            <EndTime critical=”false”> 2002-03-17T17:06:00Z </EndTime>
            <TerminationCause critical=”false”>
                <TCCode> 1016 </TCCode> <Description> normal call clearing </Description>
            </TerminationCause>
       </UsuageDetail>
    </UsageIndication>
</Message>
```

Signature

```
--bar
Content-type: application/pkcs7-signature
Content-length: 191

fHfYT64SQpfyF467GhIGfASV5jjq47n8HHGghyHhHUujhJh756tGhyHhHUujhJhjH77n8HHGTrfvbnj7
56tbB9HG4VQpfyF467GhIGbB9HGMrfvbnjn8HHGTrfvhJhjH776tbB9HG4VQbnj7567GaDJKSfYT6ghy
HhHUujpfyF47GhIGfHfYT64VQbnj756

--bar--
```

Figure 2-5 OSP UsageIndication Message

Signed inter-operator pricing and authorisation messages are also defined. Pricing allows the cost of a particular operator's service to be securely given to another operator. Authorisation is used to give permission, on a per-call basis, for use of another operator's resources. In circuit switched calls, no such authorisation exists, as every call that arrives from a connected operator is assumed to be authorised. Brokers may be used to distribute pricing and authorisation, and to clear CDRs, between operators.

An example signed OSP CDR, the UsageIndication message, is shown in Figure 2-5. The CDR is for a 3-minute basic telephony call, and contains many of the core CDR fields described in Section 2.2. The CDR is sent to an OSP broker by the terminating operator. The originating operator will generate a similar CDR, which is sent to the same broker. A UsageConfirmation record is returned by the broker indicating acceptance or rejection of the signed CDR. An OSP message consists of an Extensible Markup Language (XML) document [WWW98] and a Secure Multipurpose Internet Mail Extension (S/MIME) digital signature [Ram99] in a regular MIME [FB96] message. Use of XML will allow OSP to be easily extended. OSP messages are sent in a HyperText Transfer Protocol (HTTP) [BFN96] message, using the Secure Sockets Layer (SSL) [FKK96, DA99], described in Chapter 3, to secure the communications. TCP/IP [Com00] provides reliable transport. Use of all these message formats and protocols makes the OSP messages large and computationally expensive.

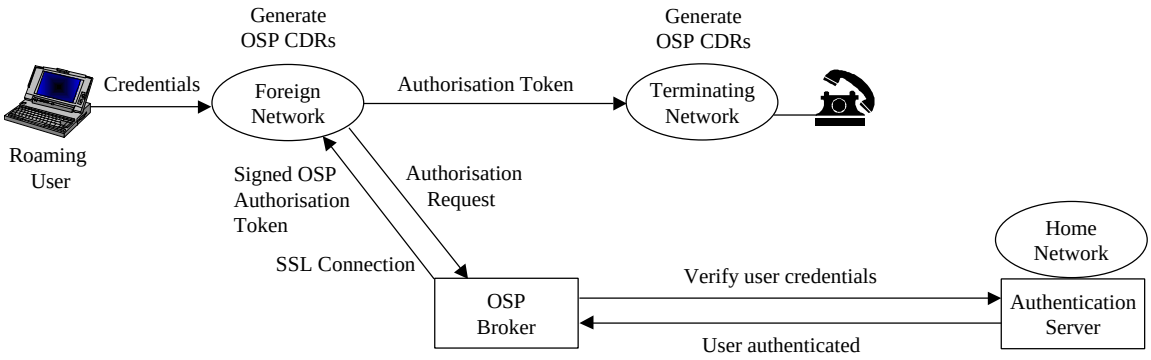


Figure 2-6 Online Authorisation for an OSP Call while Roaming

A typical call using OSP, initiated by a roaming user, will involve four online connections. The user connects to the foreign network operator, presenting his credentials. The foreign operator contacts his OSP server over SSL to obtain authorisation, as shown in Figure 2-6. In turn, the OSP server must contact the user's home authentication server to verify his credentials. Roaming user authentication is outside the scope of OSP but is examined further in Section 2.7. If the home server successfully authenticates the user, an OSP signed authorisation token is returned to the foreign network. The token is presented to the terminating operator, to complete the user call, showing authorisation for a specific amount of use. Reauthorization takes place during the call if that amount is exceeded. Upon call completion, both the foreign and terminating operators construct CDRs and clear them over SSL with the OSP broker. In turn, the OSP broker must settle with the home operator.

OSP relies on large signed messages sent online to several distant entities during a call. The signatures only serve to protect the CDRs on an open network and do not guarantee accurate or fair billing.

2.7 Billing of Roaming Internet Users

An Internet user might roam away from their home Internet Service Provider (ISP), and wish to use the services of a nearby foreign ISP. The Internet connection might be provided over a dial-up link to a network access server (NAS), or over a wireless link, and may or may not use Mobile IP [GHJP00, Per00]. Whatever

the scenario, Internet roaming poses the same problem as mobile network roaming, in that the visited network needs to be paid for the services they provide.

The IETF Authentication, Authorisation, and Accounting (AAA) working group is addressing this problem [Met99, ETSP00, AAH00, Ark99, ACGH+00]. The general AAA architecture is shown in Figure 2-7. Their approach is that the roaming user is authenticated online, through the visited network, with the home ISP. An optional AAA broker mediates messages between network providers who have no roaming agreements. The visited network is authorised, or given permission, by the home ISP to provide certain services to the user. In order to get paid, the visited network records usage consumption in CDRs, called *accounting records*, and sends them back to the home network. At a high level, the solutions are very similar to the mechanisms used in GSM and UMTS roaming, but the protocols are designed for IP network access service. However, the time taken to perform these online AAA connections can cause packets to be dropped and can thus result in noticeable service disruptions [Per00].

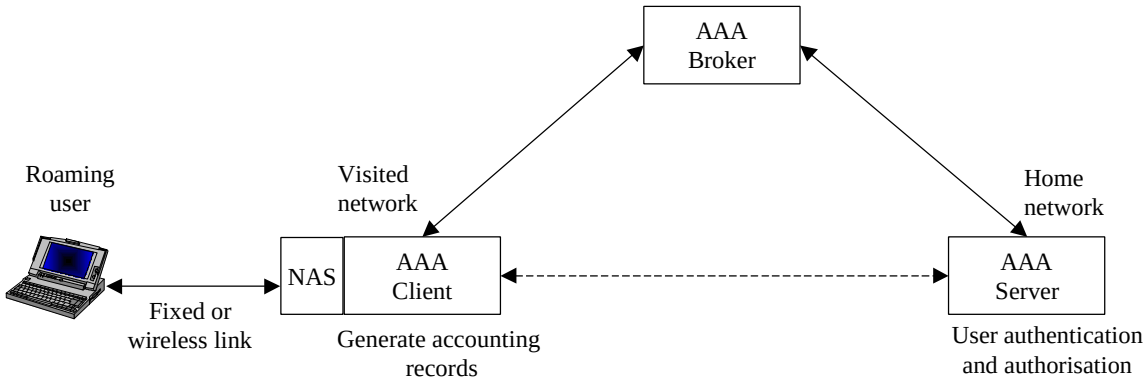


Figure 2-7 AAA Architecture

2.7.1 RADIUS

The most widely deployed AAA protocol is the Remote Access Dial-In User Service (RADIUS) [RRSW97]. The roaming user communicates with a RADIUS client at the visiting network, which in turn communicates with a RADIUS server at the user's home ISP. A shared secret, such as a password, between the user and home ISP is used to authenticate the roamer, as with GSM. The Challenge Handshake Authentication Protocol (CHAP) [Sim96] or the Extensible Authentication Protocol (EAP)[BF98] can be used to perform the authentication, although EAP is more secure as it provides end-to-end authentication and prevents replay of messages.

An *Accounting-start* message is sent from the visited network to the home RADIUS server after successful authentication. When the user disconnects, an *Accounting-stop* message containing usage information in attribute-value pairs, defined in [Rig97], is sent to the home server. The contents state the number of packets sent and received, the session duration, and the termination reason. Message integrity is provided by shared secret keys between the communicating entities. Messages can be forwarded through a RADIUS broker to prevent every network having to share a secret with every other network. Such proxying further weakens security, as the broker may tamper with the accounting records. TACACS+ [CG97] is a very similar alternative to RADIUS, where all messages are fully encrypted.

2.7.2 DIAMETER and Extensions

DIAMETER [CRAG00, Met99] is a PKI based AAA protocol that improves the security of RADIUS by using ISP digital signatures. The accounting messages [ACPZ00] are signed by the visited network and the

home ISP provides a signed receipt. This is a similar idea to the charging control algorithm of 3GPP. The home ISP authorises the user's session time, thereby limiting the extra time that could be added to fraudulent CDRs [ZC99]. DIAMETER CDRs use the MIME based Accounting Data Interchange Format (ADIF) [AL00a]. By using attribute-value pairs, where attributes are represented by a small number, it is more compact than other formats [BB00], such as OSP or IPDR.

A scheme has been proposed which replaces the online user authentication from AAA protocols with a user digital signature [Abo99]. Attribute certificates [ISO99b, FH00] issued by the home ISP, and carried by the user, can contain specific authorisations. Regular RADIUS/DIAMETER CDRs are generated but the user signature is included to prove that they were present. This approach is similar to Mini-Pay, described in Chapter 3. However, the accuracy of the CDRs must still be trusted and, without online authorisation, the user can use the authorised resources at many different visited networks, thereby exceeding their credit.

The AAA Architecture research group is defining a new generic layered protocol suite [LGGV+00, VCFG+00a, VCFG+00b] which will replace RADIUS and DIAMETER. They propose to separate the common AAA functionality from application specific modules, with a focus on authorisation. However, for many roaming scenarios, including those described in Chapter 1, the need for user authentication and authorisation, and all their associated overheads, can be completely removed if one pays in real-time.

2.8 Internet Usage Billing

With IP forming a central part of 3G and 4G proposals it seems that Internet usage will continue to increase rapidly. There will be a need to charge not only for supplying network bandwidth but also for applications and content provided over IP. The approach taken has been to define new CDRs for all types of IP traffic and applications.

2.8.1 Network Service Detail Records

Service Detail Records (SDRs) are a CDR format used to record the reservation and usage of network transport services. They are usually specific to one network type, such as IP, ATM or B-ISDN. For example, the Contract Negotiation and Charging in ATM Networks (CANCAN) ACTS project [CAN98] defined the format and contents of a SDR specifically for ATM.

The SUSIE project [SUSI99], a two year ACTS project that followed on from CANCAN, investigated charging and accounting for Internet services with Quality of Service (Premium IP services). Internet QoS, where bandwidth is reserved and the traffic is given a higher priority than normal best-effort service, can be provided using IntServ [BCS94, Whi97] or DiffServ [BBCD+98, MEH00]. A Premium IP Network Accounting Record (PIP-NAR) was designed for IntServ and DiffServ traffic.

Contents	Size (bytes)
Record description	12
Measurement point identification	16
Flow description	52
Reserved resources	30
Used resources	8
Data extension	0
Total size:	118

Figure 2-8 Premium IP Network Accounting Record Structure

The PIP-NAR contains fields describing the reserved resources and the actual volume used as shown in Figure 2-8. Due to the different parameters used by IntServ and DiffServ, two different record types were

defined, each distinguished by a record description field. A flow description with the source and destination IP addresses and the IP address of the device recording the usage is also included. The estimated cost of processing an SDR locally in real-time for hot billing is 2.17 cents [SUSI00]. This is significantly higher than the cost of processing some micropayment schemes described in Chapter 3.

SUSIE also proposed charging schemes for IntServ, DiffServ, and Multi-Protocol Label Switching (MPLS) [Li99, CDFF+99, MEH00]. They recommend *static charging* schemes to allow charges to be predicted and avoid unexpected amounts on a user's bill. As with other CDR billing systems this results in hundreds of fixed tariff plans with thousands of prices, which cannot be varied according to real-time conditions. In Chapter 5 we describe how *dynamic tariffs* can be provided if payment is made in real-time.

ETSI also investigated the data that needs to be recorded in CDRs in order to bill for IP network usage [ETSI99a]. The findings are independent of any one specific QoS protocol and are partly based on the results from SUSIE and CANCAN. An MPLS SDR is presented, consisting of connection, contract and usage information. It is over 320 bytes in length showing that the size and complexity of CDRs is increased by having to record all the reserved and used resources for a connection.

2.8.2 IP Detail Record (IPDR)

Proposed CDR formats for VoIP and IP QoS have been examined. However there are many emerging and existing IP resources and services for which the providers wish to be paid. The IPDR organisation, an industry consortium set up in 1999, is defining a generic common usage record format, the IPDR, capable of recording any type of IP resource usage. Such usage data will be generated by routers, bandwidth managers, gateways, roaming access servers, application servers, and e-commerce transactions.

The Network Data Management–Usage (NDM-U) [IPDR00] document defines the initial structure of an IPDR in XML. The IPDR contents are not fixed, and IPDR fields, called usage attributes, are defined as XML elements for different services. For example, there are 38 fields defined for a VoIP service, which are completely different than those defined for an e-mail service. Several related IPDRs, such as those reporting several HTTP requests to the same Web server, can be grouped together in an IPDR document. If the individual IPDRs have the same attributes, an XML schema can be used to state the attribute names only once in the document, making the large IPDRs more compact. Several commercial accounting and charging architectures plan to use the IPDR including Hewlett-Packard Smart Internet Usage [HP99], the NARUS architecture, and the XaCCT system.

The IPDR shows us that there cannot be a single simple CDR for IP services. Traditional billing will require that attributes be defined for all possible scenarios in every possible application and service. This will result in a huge constantly updating standard with many thousands of possible attributes. So many attributes will in turn complicate the billing process. In addition, the use of new applications will be hindered until IPDR attributes are agreed.

2.8.3 Jalda

An alternative to generating detailed CDRs is to allow the provider to calculate the charges in real-time and send a simple charge request to the billing host. The charge request only specifies the provider to be paid and the amount due from the user. Jalda [Bog00], produced by a joint venture between Ericsson and Hewlett Packard (EHPT), uses this approach to allow charging for content over IP networks. In Jalda, at the start of a session with a content provider, the user digitally signs a contract, authorising payment up to a specified amount for that provider. The signed contract is passed through the provider to a central independent billing server, with whom the user has an account. As the user utilises resources the provider sends multiple charge

requests, instead of a CDR, to the billing server. The communications are secured by SSL. The vendor must be trusted to generate fair charge requests, as with CDRs. The user signature proves their presence at the start of the session but not the amount spent. User generated micropayments, detailed in Chapter 3, can remove the unnecessary trust in the independent provider.

2.9 Emerging Problems

Telecommunications billing is moving forward from the pure circuit-switched voice domain into the realm of large-scale packet networks with variable QoS and an ever-growing plethora of applications and content. The approach has been to rigorously define new usage record formats, *xDRs*, for every possible scenario. However, this leads to a large number of different CDRs being generated in different segments of an inter-network. This is highlighted by Figure 2-9, where we show a scenario with a roaming mobile Web user. This could be a GSM roamer using WAP to browse restaurant reviews or, alternatively, to watch the latest analyst discussion on one of his portfolio holdings.

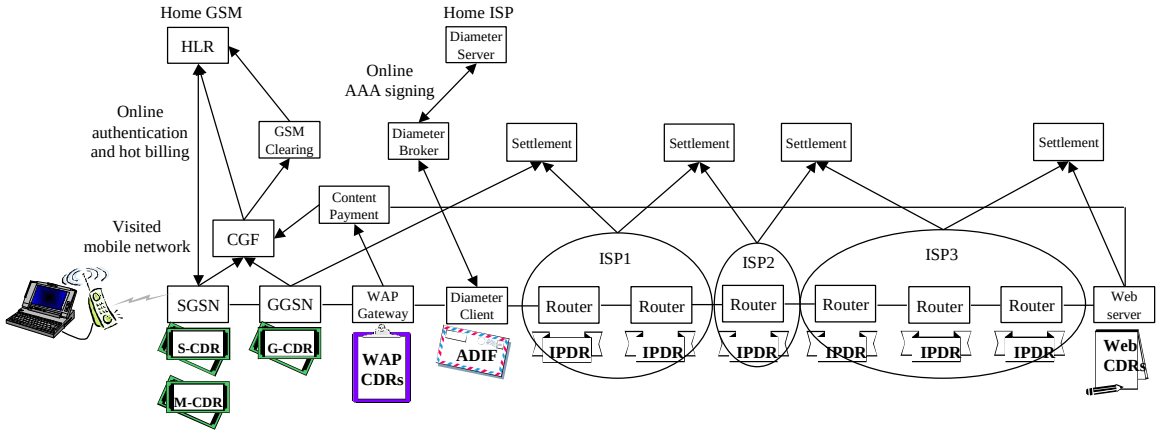


Figure 2-9 CDRs Generated by a Roaming Mobile Web User

Whatever the case, there are multiple ongoing online connections and numerous different CDR types generated. If a user has separate mobile radio access and ISP providers, as envisioned in [MH00, Hil99a, Hil99b], or if Mobile IP is employed, the online AAA protocols such as DIAMETER are used in addition to the separate mobile billing. The number of CDRs will be far greater than telephony as every state change and content request is captured. These records must be maintained for years for use in bill disputes. How billing agreements can be maintained between NOs and the huge number of VASPs is still an open issue.

The CDR approach has been in use for over 120 years. It worked adequately in monopolistic environments where a small number of trusted NOs provided a limited number of services. However, in an environment with a large number of independent access networks, where anyone connected to the network can sell their services, the CDR trust approach is riddled with inadequacies and hinders progress. In addition, static tariffs prevent real-time pricing based on network conditions. With over one billion cellular users predicted by 2004, new approaches need to be found.

Currently, the mobile plays no part in the billing process, as we show in Figure 2-9. Traditional billing allows the total amount to be paid afterwards, but cannot ensure payment or provide incontestable charging. The entire amount cannot be paid in advance since the duration of the call or the quantity of services used is not usually known beforehand. Our proposal allows the mobile to pay all the parties involved in a service by making several small payments throughout a call. In the next chapter, we evaluate existing electronic payment systems in search of techniques to advance the field to lightweight multi-party payments.

3 Electronic Payment Systems

"Everything … must be assessed in money; for this enables men always to exchange their services, and so makes society possible."

Aristotle (384 – 322 B.C.)

3.1 Electronic Payments

Since ancient times money has been used, in its many different forms, as a common measure of value. The barter system, where goods or services are directly exchanged for other goods or services, suffers from the problem that each party must have exactly what the other one wants. From simple barter arose the tendency to select one or two preferred commodity items, such as grain, salt, or gold, which were widely desired. These commodities were easy to exchange for other items and became accepted as money. A *payment* is the transfer of such monetary value from one party to another.

The history of money shows that the ways of representing value have become increasingly abstract over time [Dav96]. Cattle were used as one of the first forms of money by early farmers, as far back as 8,000 B.C. Later forms of *commodity money* included cowry shells, whales teeth, silver ingots, and gold coins. To avoid having to carry the actual commodity money, central banks later issued bank notes, which could be redeemed against deposits of gold held by them. When the issuer became highly trusted in a stable economy, it was no longer necessary to hold onto the actual commodity deposits. Such *fiat money* has value only because the trusted issuer declares it so and this is widely accepted. Recent decades have seen the emergence of *electronic money*, where the money no longer has a physical form but is represented by a series of bits, ones and zeros, in a computer system.

As money has evolved so have the methods of effecting payment. Modern payment systems are either account based or token based. Payment instruments, which effect transfer between accounts, include cheques, payment cards (credit or debit based) and bank giros. Token-based systems include cash, pre-paid phone cards, and postal stamps. An *electronic payment system* or *network payment system* allows monetary value to be transferred from one entity to another across a computer network. With the emergence of the global Internet many different electronic payment systems have been proposed in recent years [OPT97, AJSW97]. Many of these are based on existing payment instruments while others introduce new forms of value representation and exchange.

The purpose of this chapter is to review existing payment techniques and the underlying cryptographic algorithms upon which they are built, in order to access their suitability for use in a multi-party payment environment. The chapter is organised as follows. In Section 3.1.1 the distinction is drawn between macropayment and micropayment solutions, and a generic network payment model is introduced. An overview of existing macropayment systems and research is then presented. An examination of the primitive constructs and techniques used in these electronic payment systems is necessary to understand why they are

completely unsuitable for frequent multi-party payments, as envisioned in Chapter 1. A comprehensive review of micropayment research is then presented, and the systems are grouped based on the cryptographic primitives used. An extensive micropayment survey has not yet been published, and it is necessary in order to understand the different efficiency trade-offs made in each scheme and their resulting unique properties. Based on the observations made during the survey, we contribute three new proposals which improve on the state-of-the-art: a hash chain which can be infinitely extended; an efficient method to derive a hash chain authentication tree; and an improved shared secret micropayment scheme with user fairness. These new micropayment innovations are independent of our multi-party micropayments in later chapters. The lessons learnt from examining micropayments are summarised in Section 3.4. Finally, attention is drawn to issues that have not yet been adequately addressed by existing micropayment research.

3.1.1 Macropayments and Micropayments

The majority of existing electronic payment systems were designed to allow payment to be made for tangible goods with little chance for fraud. The security of these systems, implemented using cryptography and/or secure hardware, must be strong enough to prevent fraud or to detect the party responsible after it has occurred (post-fact detection). Such electronic payment systems, designed to securely allow payments ranging in value from approximately one Euro to several thousand Euro, are known as *macropayment* systems. Payments larger than this are usually effected using traditional bank transfers over private banking networks.

Macropayment schemes are modelled on real world payment instruments. In the next section, an overview is presented of the macropayment systems based on credit cards, cheques and cash. Each of these macropayment instruments have a minimum transaction overhead, usually imposed by the issuing bank, which prevents them being used for payments of a few cents. A second prohibiting factor is their heavy use of computationally expensive cryptographic operations, such as public key cryptography. In Chapter 4 a quantitative analysis of the computational overhead of several example systems is investigated. It is shown how these factors make macropayments too inefficient for repeated frequent transactions such as making a payment per second for a telephone call.

In contrast, *micropayment* solutions are designed to allow the efficient frequent transfer of very small amounts, perhaps less than a cent, in a single transaction. The increased efficiency is obtained by slightly relaxing the security, which is acceptable due to the small amounts involved. The computation required to falsely obtain value in a micropayment system is made more expensive than the actual value that can be obtained. Micropayment systems are examined in detail in Section 3.3.

3.1.2 Network Payment Model

Electronic payment systems involve three or more independent entities that communicate across a network. To make a *direct payment*, a payer, or user/customer, sends an electronic payment instrument to a payee, or merchant/vendor, across the network. The majority of consumer network payment systems use direct payments. In an *indirect payment*, the payment is effected through a financial agent, without online communication between the payer and payee. This form is more common in business-to-business electronic payments.

In both cases, the financial agent is the authority which vouches for the value of a payment and exchanges money from an existing form to its electronic form and vice-versa. The financial agent may be split into two separate financial institutions as shown in Figure 3-1. One financial institution is the issuer, which provides the payer with the payment instrument necessary to make a payment. The other is the acquirer, which accepts

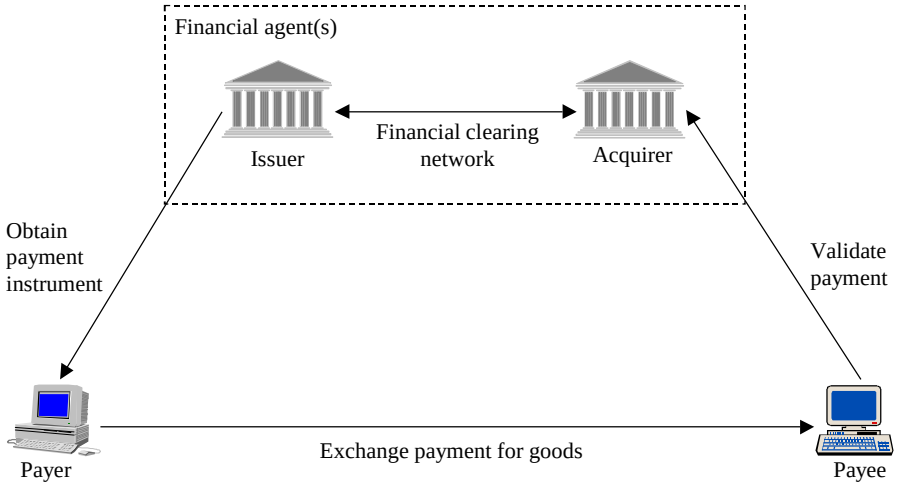


Figure 3-1 Direct Network Payment Model

the payment instrument from the payee and redeems it for the specified monetary value. Dedicated financial clearing networks, such as the Automated Clearing House (ACH) network, the Clearing House Interbank Payments System (CHIPS), the Bankers Automated Clearing Service (BACS) in the U.K., and the Society for Worldwide Interbank Financial Telecommunications (SWIFT) network, are used for payment communications between financial agents. The payer and payee are usually connected over open networks.

Both macropayment and micropayment systems can be represented by this basic model of electronic payment. In some systems, such as those using stored value cards, the payee can use the payment received to make another payment, without contacting the financial agent. In this way, money can flow from entity to entity, as with paper cash, only being returned to the acquirer when one payee wants to exchange their value. An overview of macropayment protocols is now presented, followed by a detailed examination of micropayments.

3.2 Macropayments

Electronic payment protocols based on payment cards, cheques, and cash are now examined. Particular attention is drawn to the performance bottlenecks of each system. At first, electronic cash seems to be ideal for paying for mobile communications and services – electronic coins can be released as resources are consumed. In order to highlight its complete unsuitability, a closer look at the complex constructions and techniques used in the many different forms of electronic cash, is given. Smart cards may be used to add security and flexibility to a payment system and the variety of schemes employing these chipcards are discussed in Section 3.2.4. The overview of macropayments is concluded by describing how the payment techniques discussed have been adopted for mobile commerce solutions.

3.2.1 Payment Card Systems

Credit card and debit card based schemes are the most widely used means of conducting a consumer purchase over the Internet. The Secure Sockets Layer (SSL) [FKK96], and its successor, Transport Layer Security (TLS) [DA99], are used to encrypt all messages, including the payment card details, sent between the payer and the payee over the World Wide Web. A WAP version of TLS, the WTLS [WAP99a], is used with wireless links from WAP devices, and has been integrated with secure hardware [WAP99b]. SSL uses authentication based on asymmetric (public key) cryptography [DH76, Sch96], with each entity requiring an X.509 certificate [ITU97a, HFPS99], issued by a trusted third party. Normally only authentication of the vendor (payee) takes place. After authentication all messages are encrypted using symmetric cryptography.

The actual cryptographic algorithms used are negotiated at connection setup. The payee verifies the card details, at the time of purchase, through an existing financial network.

However, such secure transport schemes have several disadvantages. The vendor has full access to the card details, and this has been shown to be a large source of fraud [Wal99b]. In addition, without strong payer authentication it is not possible to know if the payer is actually the payment card owner, as the card details may have been stolen. Indeed, Internet vendors report a repudiation level ranging from 25% to 30% of their total revenues, according to Jupiter Communications. Several Visa member banks have experienced fraud rates as high as 50% for Internet transactions [Tri00].

A number of solutions, which address these problems, have been proposed [Sir97]. Often these schemes use a payment gateway to relay messages between the public Internet and the private financial clearing networks. The CyberCash credit card payment scheme [EBCY96, CBHL95] and Internet Keyed Payment Protocols (iKP) [BGH+95] are two such schemes. Each cardholder is issued with a digital certificate which links a public key to the card details. The private key is used to generate a digital signature which can be verified with the certificate and show the cardholder to be genuine. The card details are encrypted with the public key of the payment gateway and forwarded through the vendor. Only the payment gateway's private key can decrypt the card details, and hence they are not available to the vendor.

Secure Electronic Transaction (SET) [MV97] uses the same model and is the proposed standard for payment card purchases made over open networks, from MasterCard and Visa. It offers an improvement over earlier schemes by using a Certification Authority (CA) hierarchy instead of a single CA, stronger public key encryption of the card details, a stronger card blinding mechanism [Kra99], and a dual signature scheme [OPT97, MV97] to link order and payment details together without revealing these details. However, SET is a heavyweight protocol requiring a large number of computationally expensive signatures and messages to complete a single transaction. Lightweight versions of SET have been proposed which reduce the computational time by up to 53% and the communications overhead by up to 80% [HZI98]. Versions have also been designed for mobile networks [WZ99, RS98] and for smart cards [SET99, SET98, BG97].

These payment card schemes are *online* in that a network connection must be established to a third party, the payment gateway/acquirer, during the purchase phase. This is done to verify the card details and check the database of blacklisted stolen cards. This adds an extra communication overhead and the third party can prove to be the bottleneck in the system, as it is constantly contacted by all active vendors. Mu and Varadharajan [MV98b] propose three related credit card schemes that allow the vendor to verify the purchase *offline*, with no third party network connection. In their schemes, the user proves knowledge of the card details to the vendor, without revealing those details. This is performed using a *proof of knowledge* based on the equality of logarithms [CP92a, VvT97], which is passed to the vendor. The proof of knowledge constitutes proof of payment, and is later sent to the issuing bank to complete the transaction. However, since the schemes are offline, they do not protect against stolen private keys and card details being used to perform a large number of fraudulent transactions.

In the above schemes, the cardholder is identified to the vendor either by the card details or a digital signature. Some privacy is present in that network eavesdroppers or the financial institutions may not know the order details. Efforts have been made to allow anonymous credit card transactions [LMP94, LMP96] and anonymous funds transfer [KLM94] where the customer can remain unknown to the vendor, and the vendor remains unknown to the bank. These schemes use information separation, where the information needed for a transaction is separated into different components and hidden, using cryptography, from those parties that do not need to see a specific component. The schemes rely on trusted payment switches to anonymously

relay messages, thereby increasing the communications cost. The frequent use of temporary public keys greatly increases computational overhead. The anonymity is broken when parties collude to share transaction details. Mu and Varadharajan [Mv98a] propose an offline anonymous credit card scheme that uses an anonymous user public key certificate, issued by a bank. The authenticity of the anonymous certificate is based on an equality proof of knowledge [CP92a] to the vendor, where the user proves ownership of the certificate and the blinded credit card without revealing their identity. However, the user is only anonymous to the vendor and the purchase requires an additional proof of knowledge. In addition, the same "anonymous" certificate is used for all transactions, which allows them to be linked, even if the bank does not reveal the identity of the cardholder.

3.2.2 Electronic Cheques

A cheque is a signed order to pay an identified payee using funds from the payer's bank account. In an *electronic cheque* scheme the cheque is usually generated and digitally signed by the payer before being passed across the network to the payee for verification. The payee endorses the cheque by applying a further digital signature before sending it to his network bank. Existing financial networks can be used to clear the electronic cheque between the payer and payee's banks. To ensure the availability of funds during a purchase, the cheque should be cleared online. However, the digital signatures ensure that each party is fully identified, which allows an offline solution with post-fact detection.

The Financial Services Technology Consortium (FSTC) has implemented an offline electronic cheque scheme [And98] using tamper-resistant smart cards. The smart card holds a user's private signing key, allowing portability and increasing security against theft and misuse. Both payer and payee require smart cards. Cheque numbers with nonces guarantee the uniqueness and freshness of an e-check. Further details of the smart card functionality and its integration with the Web can be found in [JB95]. The MANDATE II project created a similar e-check scheme [Man98] with a smart card based "chequebook" which holds two public/private key pairs, one for signing and the other for encryption. The scheme is unique in that it only allows a cheque to be transferred to another single smart card by encrypting it with the public key of that card before releasing it. This is used to prevent double encashment of cheques by the payee.

Flexicheck [Tew95] is another offline cheque-like scheme. Users prepay their bank for the ability to issue bank-signed certified cheques up to a specific total value, from their smart card. The scheme is novel in that it uses control vectors [Mat91] to restrict usage of the bank's signing key, which is held on the card. Flexicheck uses similar security techniques as stored value cards, described in Section 3.2.4, and suffers from the same vulnerabilities of relying completely on the tamper resistance of the hardware. In contrast, NetCheque [NM95] is an online cheque scheme that uses only symmetric cryptography, for computational efficiency. It is based on the Kerberos system [NT94] and an extended Kerberos ticket acts as a digital signature. However, use of shared secrets and online Kerberos authentication servers make it more suitable for local area solutions rather than a global system.

Electronic cheques have a similar payment model to payment card schemes. A signed payment order (cheque or credit card) is transferred through the payee to an acquiring bank for authorisation and clearing with the issuing bank. Like online credit card schemes, digital signature creation and verification is required by all parties. Money is taken from the payer's account at or after the time of purchase, except in the case of a prepaid certified cheque.

An indirect payment alternative to electronic cheques and payment cards is simple *account transfer*. The payer authorises funds to be transferred from one account to another, usually at the same network bank. If anyone can hold such an account this allows not only customer-vendor payments but also person-to-person

payments. Accounts can be funded using a credit card payment or, alternatively, by purchasing a physical prepaid card. The card number maps to an account with the prepaid amount, as with prepaid mobile phone cards. SSL is used to protect any communications with the bank and authentication is usually only password based. The vulnerability of simple password schemes was illustrated in a recent attack on a prominent account transfer scheme where thousands of passwords were stolen [Sul00]. Commercial examples of account transfer schemes on the Internet include InternetCash, Yahoo PayDirect, PayPal, BillPoint, Mon-e, ProPay, and RocketCash. The disadvantage of these schemes is that all participants must have an account at the same central server. In addition, both parties need to be online with the bank server to make the payment and verify that it is received.

3.2.3 Electronic Cash

Physical cash is token-based fiat money, where the value tokens are coins or bank notes. The tokens are produced in such a way that it is easy to verify them as being genuine, perhaps through a watermark or metal strip in a bank note, but they are very difficult to forge. The tokens represent the actual monetary value and transfer of them completes a payment, without any transaction fees. If the current holder destroys the tokens then she has lost the value they represent. Cash allows payer *anonymity* and payment *untraceability*, since there is no information on the tokens to link the payment to the payer, unlike payment card and cheque transactions. A bank could conceivably record the unique serial numbers on a bank note when it is withdrawn, in the hope to discover who later deposits that note. However due to the *transferability* of cash, in that it can be passed from person to person in payment indefinitely, without being returned to the bank, it is impossible to trace its path without the co-operation of each party it passed through. Coins do not have such serial numbers and cannot be traced.

The term electronic cash is often applied to any electronic payment system that appears to offer any of the above attributes of physical cash. We prefer to think of *electronic cash*, or *digital cash*, as a direct electronic macropayment system, where the payment instrument consists of prepaid payee-independent electronic value tokens issued by a trusted financial agent. This definition excludes account-based systems that rely on the authenticated transfer of value between accounts. Unlike electronic value tokens, if the account transfer message is destroyed, value is not “lost” by the payer. The definition also excludes bank-certified electronic cheques, which are vendor dependent, in that the payee is identified in the cheque before payment occurs.

An electronic cash system consists of a withdrawal, payment, and deposit phases. We now describe how these steps work in a simple non-anonymous electronic cash system. A user Alice withdraws electronic coins of specific denominations from her online bank. A coin consists of a serial number, to uniquely identify it, and the coin denomination. Each coin is digitally signed by the bank to show that it is authentic. To make a payment, Alice assembles the correct amount in coins and sends them, across a network, to the payee Bob. Bob can verify the bank’s signature on each coin. To ensure that these coins have not been spent before, Bob deposits them in the issuing bank, for verification. The bank maintains a database of the serial numbers of all spent coins, to prevent double spending of tokens. If the coins have already been spent then their serial numbers will be present in the database. Otherwise, the payment is valid, and the serial numbers are then entered into the database, having now been spent.

The above system lacks some of the properties of physical cash such as anonymity and transferability. Obvious shortcomings include the fact that the user must have the exact change to be able to make the payment, and that the bank must be contacted online during each purchase to prevent double spending. Recently, much work has been done to extend and improve on this basic scheme. To understand the capabilities of current electronic cash proposals, and the corresponding communications and computational cost, we now highlight the important results.

3.2.3.1 Anonymity and Untraceability

Providing anonymity has been one of the main motivations of electronic cash research. Different levels of payer anonymity are possible with different electronic payment systems. Payee anonymity is not considered as consumers often require a payee signed receipt. Many schemes offer *privacy* of the payment, where the transaction details are hidden from network eavesdroppers through use of encryption, such as with purchases made using SET or SSL. Payer *anonymity to the payee* allows the identity of the payer to remain unknown to only the payee, but not the financial agent. Our earlier example of a simple electronic cash scheme provides this in that Bob receives anonymous coins from some payer across the network. However, by noting who deposits the serial numbers issued to Alice, the bank will discover the payment path.

This anonymity can be extended by allowing any holder of valid coins to anonymously exchange them for new coins with the bank. Since the bank cannot know whether it is the payer or payee performing the exchange, it cannot know the transaction path. Our own electronic cash system PayMe [PO95], which was designed to provide a secure and scalable method of effecting payment using the World Wide Web, and the earlier NetCash [MN93], provide this level of *limited anonymity*. It is limited in that if the bank refuses to allow coin exchange, or if the payee colludes with the bank, then the anonymity is removed.

Full anonymity, or *untraceability*, is where the identity of a payer in a particular transaction remains unknown to all parties, even if those parties collude to swap information. Full anonymity can be accomplished by getting the issuer to sign the electronic coins during withdrawal in such a way that she cannot see the serial numbers on the coins she is signing. This can be done using a *blind signature* [Cha82, CB97], where the coin is "blinded" by the user by applying a random quantity known as the blinding factor. The bank signs the random-looking blinded coin, and the user can later remove the blinding factor, leaving a valid signed coin whose serial number is unknown to the bank. The bank is now unable to link a specific withdrawal with a specific deposit. This provides stronger anonymity than physical cash. The bank uses a different digital signature key for each coin denomination to prevent it unintentionally signing an arbitrary message. A final aspect of anonymity is *unlinkability*, where two coins from separate transactions cannot be linked as having been withdrawn by the same user.

Blind signature schemes have been devised for different public key algorithms with performance based on the efficiency of the underlying algorithm. Blind signatures based on RSA signatures [Cha85] and Schnorr signatures [Sch89] have been used in electronic cash schemes [CFN88, CP92a, Bra93, Fer93a], including the commercially deployed Ecash [Sch98]. Variants on these include Ferguson's randomised blind signature [Fer93a], the Chaum-Pederson "double Schnorr" signature [CP92a], and Chaum's blind unanticipated signature [Cha87, Cha88]. The formal security of blind signatures is investigated in [JLO97]. It is worth noting that, for the same level of security, the bit length of Schnorr signatures is less than half the length of RSA signatures. Also since the security of Schnorr signatures is based on the discrete logarithm problem [Ros93, RSA00] they can be implemented in any setting where a counterpart of the discrete logarithm problem is difficult. Hence Schnorr blind signatures can be implemented using elliptic curve cryptography [JM97], which is more efficient in terms of communications and storage than other non-elliptic curve based public key algorithms, as we show in Section 4.2.5.

3.2.3.2 Double Spending: Prevention and Post-fact Detection

Since electronic coins are a series of bits a user might copy them and attempt to spend them a second time at a different vendor. There is no cryptographic method to prevent this. The only way to guarantee prevention of *double spending*, or *multiple spending*, is to check the database of spent coins online at the issuer at the time of purchase. The size of this database can be limited by setting coin expiry dates, by limiting the lifetime of coin signing keys. PayMe [PO95] improved the scalability of the database by recording every coin in

current circulation rather than every coin ever spent, at the cost of providing limited anonymity rather than full anonymity.

To reduce communications costs it is desirable to make an electronic cash scheme offline. A tamper-resistant smart card can be used to hold coins and prevent double spending in an offline solution, as discussed in Section 3.2.4. However, if the security of the smart card is compromised, there needs to be a method in place to reveal the identity of a double spender.

In an offline solution where use of a tamper-resistant device is not desirable or possible, a similar method of post-fact detection is required. To be able to detect the identity of a double spender the coin needs to include, or be linked to, this identity information. In a non-anonymous system it is straightforward to include user identity information in the coin. However, in a fully anonymous cash system, the identity information must be split in such a way that one piece reveals nothing about the payer, but two separate pieces completely reveal her identity.

The split, or fragmented, identity information is created and linked with the coin during the blind signature based withdrawal. It is done in such a way that the bank can verify its presence and correctness without being able to see its contents. When the payer spends the coin, she is forced to reveal one piece of the identity information. A challenge-response protocol is used to ensure that the piece selected is random, and not controlled by the payer; otherwise the payer could always release the same piece. When the payee later deposits the coin offline, the identity piece is also included. If the same coin is spent twice, two different identity pieces will eventually be deposited and the double spender will be revealed.

3.2.3.3 Cut-and-Choose and Single-Term Systems

Two techniques have been proposed in the literature for binding the split identifying information to a coin. These are the cut-and-choose method and single-term systems.

With the *cut-and-choose* technique, the user identity is represented by a pair of large numbers. One number of the pair reveals nothing but the two numbers together provide the identity. During withdrawal, the user constructs and blinds *2n* different pairs of numbers. The bank randomly chooses *n* of these blinded pairs and gets the user to unblind them so that they can be verified. If all *n* pairs are valid it is likely that the other *n* blinded pairs are also valid and so the bank signs and links these with the electronic coin. During a purchase, the payee Bob sends a challenge consisting of *n* random bits. For each bit the corresponding piece, the first piece for a 0 or the second piece for a 1, of the *n* pairs is released. Since Bob only has one piece of each pair, no identity information is revealed. The challenge response is deposited with the coin. If the payer Alice cheats and spends the coin again, a different random challenge will be issued and different pieces will be revealed. With a large enough value of *n*, it is very likely that different pieces for at least one pair will be released in each payment. The bank can then combine the two pieces to reveal the double spender. The cut-and-choose scheme was used in the first offline electronic cash proposals [CFN88]. However, it is not efficient or practical because each coin is accompanied by *n* pairs of large numbers which greatly increase the storage and communications overhead. The protocols are also interactive, requiring three or four messages for each action.

Recent electronic cash proposals have reduced the *n* pairs needed in the cut-and-choose technique to only one term or item and are hence known as *single term* systems. They rely on zero knowledge proofs [MOV96]. Each user creates a public key pair in such a way that the secret key reveals the user's identity. During withdrawal the user Alice proves knowledge of the secret key without revealing it. The public key is linked to the blinded coin. To make a payment Alice proves to the payee Bob that she holds the secret key

corresponding to the public key linked to the coin, again without revealing it. However the zero knowledge proof is done in such a way that two proofs for the same coin will reveal the secret key, and hence Alice's identity. While single term systems reduce the storage, communication, and computational cost by a factor of *n*, they are only as efficient as the zero-knowledge proof, which uses asymmetric cryptography.

The Chaum-Fiat-Naor scheme [CFN88] was the first fully anonymous offline electronic cash scheme proposed. It uses blind RSA signatures and the inefficient cut-and-choose technique to include identifying information. Okamoto and Ohta [OO89] modified this scheme by removing the cut-and-choose technique from every withdrawal and instead performing it once when setting up a new user account. The user is given an untraceable "license" which is used in each withdrawal. However the coins are *linkable*, as two coins from separate transactions can be linked as having been withdrawn by the same user. Thus, if the user's identity is revealed for one transaction, it can be discovered for all transactions.

Brands proposed an offline electronic cash scheme [Bra93], based on blind Schnorr signatures, which improved on the above schemes. It is more efficient because it is a single-term system and the Schnorr signature generation can be implemented efficiently, as mentioned in Section 3.3.3.4. Ferguson proposed another single-term system [Fer93b] by modifying the normal blind RSA signature to produce a randomised blind signature. Both the payer and the bank add random data to the coin in order to prevent the payer from breaking the single term identification scheme. However, Ferguson's proposal is more complicated than earlier solutions and less efficient than Brands' scheme.

3.2.3.4 Transferability

An electronic coin is *transferable* if a payee who received the coin in payment can use it to make another subsequent payment. Transferability is desirable in that it reduces the communication needed with the issuing bank, although this is more important for physical cash. The electronic cash schemes mentioned above are not transferable as each coin can only be spent once before being returned to the bank. A transferable coin must contain the blinded identity information of all the users that owned and spent that coin, in order to identify double spenders. Hence a transferable coin will grow in size each time it is spent, as shown by Chaum and Pederson [CP92b]. This makes each successive payment more inefficient. The size of a coin can be limited by setting a maximum number of allowed transfers. Tewari, O'Mahony, and Peirce [TOP98] proposed a transferable ecash scheme that uses transaction lists and splitting of identity information.

Transferable coins are more open to abuse as each successive owner can double spend. The time a coin is in circulation is much greater, allowing more abuse before the double spending is detected when the coins are finally deposited. The payee is also anonymous in a transferable system, preventing any transaction record and creating deficiencies similar to physical cash, such as money laundering and tax evasion.

3.2.3.5 Divisibility

One problem with electronic cash is that, to make a payment, the user needs to have the exact amount in the correct coin denominations. This requires carrying a multitude of coins of different denominations, which will increase storage costs and result in the user holding more cash than is desirable. To avoid this, the user can withdraw the required coins at the time of purchase when the exact amount is known, but this will cause the system to be online. Another option is to have the payee issue change to the user. However, this just moves the problem of exact coins from the payer to the payee. It can also break the anonymity since the vendor will know the serial numbers on any change he issues and, by colluding with the bank, can discover who deposits the change.

To overcome these problems, divisible coins were proposed by Okamoto and Ohta [OO91]. A *divisible coin* is an electronic coin that can be divided into coins of smaller denominations whose total value is equal to the value of the original coin. Blinded identifying information is included with the divisible coin in such a way that if a user overspends the total value for the coin her identity is revealed. The schemes work by associating a binary tree, whose nodes represent different denominations, with each divisible coin. However, the split coins, or sub-coins, are linkable in that they are known to come from the same original coin. The Okamoto and Ohta scheme [OO91] is based on the inefficient cut-and-choose technique. Eng and Okamoto [EO94] improved on the efficiency by basing their scheme on Brands' single-term electronic cash. Okamoto [Oka95] proposed another divisible scheme using single-term coins, where the bulk of the computation takes place during the once-off account establishment. However, Chan et al. [CFMY96] identified, and fixed, a flaw in this scheme which allowed users to overspend without detection. Tsiounis [Tsi97] improved the efficiency of the account establishment step by three orders of magnitude. The scheme has comparable efficiency with non-divisible off-line electronic cash.

In his Ph.D thesis, Tsiounis [Tsi97] presents an algorithm allowing a pre-determined minimum amount of k exact payments to be made for a specific withdrawal amount N. He shows that, for certain values of k and N, keeping multiple non-divisible coins is more efficient than using a single divisible coin, due to its complex structure. He uses this result to emulate unlinkable divisible cash, using a carefully chosen collection of non-divisible unlinkable coins.

3.2.3.6 Revoking Anonymity

Adoption of electronic cash by governments and financial institutions has been slow, as full anonymity may not be desirable for law enforcement purposes. To satisfy possible legal interception requirements it is necessary to be able to remove the anonymity when required. Electronic cash schemes that allow the anonymity to be removed by a Trusted Third Party (TTP) are known as *fair electronic cash* schemes. Similar anonymity revoking schemes have been proposed for encryption and signature systems [FY95, SPC95], and are referred to as *key recovery* or *key escrow* services.

Stadler, Piveteau, and Camenisch [SPC95] first proposed a method for allowing traceability with electronic cash schemes based on the cut-and-choose technique. A number uniquely identifying the withdrawal is encrypted with the public key of the TTP and included in the electronic coin. The withdrawal is performed with online interaction with the TTP. If required, the deposited coin can later be given to the TTP, who can decrypt the withdrawal number and identify the payer at the issuing bank. A mechanism is also provided for tracing where coins from a specific withdrawal are spent. Schemes that improve on this by keeping the TTP offline until tracing is required include [FTY96, DFTY97, Tsi97], [CMS96], [JY97], and [Mra96]. However, all these schemes add an additional computational overhead that makes them less efficient than untraceable electronic cash.

When the cryptographic techniques used are considered with the performance measurements of Section 4.2, it can be seen that electronic cash has extremely poor performance. In addition, many of the protocols have undesirable properties – the post-fact detection of cheaters and use of global blacklists; problems with failed transactions; fault tolerance; and lost money [XYZZ99, PW97]; legal problems of full anonymity such as money laundering and tax evasion [SN92]; and the difficulty of detecting a security failure or forged coins [ENK99, EN98].

3.2.4 Stored Value Cards and Electronic Purses

The offline electronic cash schemes discussed above can detect an anonymous double spender after the fraud, but cannot prevent it occurring. In a real-world scheme with a large number of users, it is more practical to

prevent the fraud occurring rather than trying to expel the fraudsters. A *tamper-resistant* device [EGY83] is constructed so that it is very difficult to change data stored within it. In addition, tamper-resistant devices provide secure protection for user private keys, and can allow portability of this data. An *electronic purse* is a tamper-resistant device, such as a smart card, that is used to securely store electronic cash securely and prevent double spending. We reserve the term *electronic wallet* to refer to the software, present on a user device, that implements the user functionality of an electronic payment system.

The increased security of smart cards will prevent small-scale abuse. However, determined adversaries are likely to be able to overcome the tamper-resistance, as illustrated by [AK96, BDL97, BS97]. Such schemes must not be considered tamper-proof, and require additional security mechanisms as present in traditional electronic cash. Further details of smart card technology and security are available in [Fan97, Hen97, RE97]. However, due to their limited computational and storage capabilities, use of smart cards adds a significant delay to the overall transaction time. Even with cryptographic co-processors, algorithm speeds are considerably slower than conventional workstations, as illustrated in Section 4.2.4.

In addition, the user must trust the smart card, as they cannot monitor exactly what data is being transmitted by it. To remove this need to trust the device, Chaum and Pederson [CP92a] proposed electronic purses with observers. The *observer* is a tamper-resistant device embedded in user hardware such as a hand-held computer or mobile phone. The observer is required to complete a transaction, and it can prevent the user cheating. All traffic from the observer can be monitored by the user device and therefore does not need to be trusted. Observers have been incorporated into several offline electronic cash schemes [Fer93b, CP93], with Brands' observer scheme [Bra93] being the most efficient. The European ESPRIT project CAFE (Conditional Access for Europe) implemented an anonymous offline electronic cash scheme using an observer based electronic purse [BBC+94]. Electronic purses with their own user interface, such as a keypad and display, have the additional advantage of allowing user PINs to be entered directly, so that this information cannot be obtained and abused by untrusted terminals.

A *stored-value card* maintains a temporary account balance, rather than actual electronic tokens, on a smart card. The card securely holds trusted bank keys, which are used to digitally sign messages passing between cards. Value transfer occurs when the payer's card sends a signed payment instruction to the payee card or terminal. Having received an acknowledgement, the payer card will decrement its balance counter by the specified amount. While such schemes provide some of the properties of cash, we exclude them from our definition of electronic cash as explained in Section 3.2.3. As noted by Yacobi [Yac97, Yac99] it is easier to control fraud in a smart card using electronic coins than in a stored value card.

Mondex [Eve97], which is a subsidiary of MasterCard, is an example of a pre-paid stored value card, which allows card to card payments. While the technical protocols are proprietary, it does use some form of public-key technology to generate a signature on inter-card messages. Electronic purse schemes, which only allow card to vendor-terminal payments, include Proton, Geldkarte, and VisaCash. Unlike Mondex, the issued value does not originate from a single source bank and, during the clearing process, each payment must be settled on an individual basis between the acquiring and issuing bank.

At the time of writing there are over twenty different stored value card schemes in use in Europe. The Common Electronic Purse Specifications (CEPS) [EV99], from Visa and Europay, will allow interoperability of electronic purses. It specifies that RSA and DES will be used in offline transactions. However, card-to-card payments are not possible with CEPS. The earlier Europay MasterCard Visa (EMV) standard [EMV98, EMV00] provides common specifications for communications between cards and readers for credit/debit

card applications. MasterCard have implemented an EMV compliant credit/debit application, called M/Chip [Mas00]. A report detailing the deployment of smart card payments in Europe can be found in [ESTO99].

3.2.5 Mobile Commerce Payments

The widespread use of mobile phones has given rise to a number of schemes to allow the purchase of goods from these mobile devices. At the simplest level, a special vendor-specific phone number is called from the mobile phone, which causes a predefined amount to be billed to the caller's telephone bill. Such schemes can only be used for local vending machines or terminals, are restricted to a single payment amount, provide limited security, and require the user and vendor to share the same mobile operator.

The more advanced schemes are the same as payment card or stored value card systems, except that the user transport connection is now over a wireless link. Examples include the Nokia/MeritaNordbanken/Visa mobile SET solution, virtual credit cards from Trintech, MobilSmart from Logica, Mammom mobile wallet from Telenor, Ericsson's Mobile e-Pay [Kal00], and the Fundamo account-based scheme. Recently, several industry forums were created to develop standards for mobile commerce based on existing wireless solutions. These include the Global Mobile Commerce Forum, Mobey, Mobile Electronic Transactions (MET), and Radicchio.

Some schemes propose the use of agents [RS98] and server-side wallets [WZ99] to move the performance burden from the mobile device into the network. However, all the systems still inherit the same scalability and performance problems of traditional macropayments; namely, online third party connections and expensive cryptography for every single transaction. Furthermore, new delays and problems are introduced by the limited wireless link. In the following section, micropayment schemes are presented that alleviate many of the bottlenecks observed in macropayments.

3.3 Micropayments

A *micropayment scheme* is an electronic payment system designed to allow efficient frequent payments of small amounts, as little as a tenth of a cent. In order to be efficient, and to keep the transaction cost very low, micropayments minimise the communication and computation used. The previous section showed that the majority of macropayments use online communication with a third party and/or employ computationally expensive public key cryptography. In contrast, micropayment schemes aim to allow offline payment verification using lightweight cryptographic primitives. The security requirements are relaxed, in order to increase efficiency, which is acceptable due to the small amounts involved. The cost of fraud is made more expensive than the possible value to be gained by cheating, whereas with secure macropayments fraud without detection is made almost impossible.

A number of macropayment systems allow inefficient payments of small amounts. We define a *small value payment system* to be a macropayment scheme capable of allowing infrequent payments of small amounts, which does not scale to frequent micropayments. The performance analysis in Section 4.3 highlights this difference.

The majority of micropayment schemes were designed to pay for information goods on the Internet. A network user might pay to consult an online database, read some financial Web pages, listen to a song, or play an online game. As envisioned in Chapter 2, we see the potential to pay not only for information but for voice and data transport services, and the quality of service provided. Ultimately, it should be possible to pay the multiple parties involved in providing all aspects of a service as that service is consumed. The demand for micropayments is illustrated by the fact that Visa estimates that, worldwide, over 1.8 trillion dollars is spent in transactions of less than 10 dollars every year.

Following the surge in macropayment research, many micropayment schemes have now been proposed. We have classified micropayments into six categories based on the cryptographic constructs used and the communications overhead – online, public-key based, hash chains, hash collisions and sequences, secret sharing, and probability based. We argue that the first two categories, online and public key based, are small value payment schemes rather than pure micropayments. In each category the key concepts are explained and analysed and the important schemes referenced. From this a detailed performance analysis is presented in Section 4.3. To our knowledge no other such micropayment survey and comparison has been published.

3.3.1 Online Small Payments

A number of schemes, which contact a trusted third party online during every payment, have been put forward as micropayment solutions. We do not consider these schemes to be scalable enough to handle many repeated small value payments. While the online communications with a central party will quickly become the bottleneck in the scheme, as well as doubling the latency, some of these schemes do reduce the computational cost over traditional macropayment systems.

3.3.1.1 Online Shared Secret Keys

CyberCoin [Cyb97], Clickshare [Oli96], and Tang's scheme [Tan95] are examples of online small payment schemes which use only symmetric cryptography. Users and vendors share unique secret key material with the broker. The customer's secret key is used to authorise a payment order, including an identifier of the item being purchased. Since the vendor does not share the customer secret, he cannot check the payment order, or forge it, and must verify it online with the bank. Tang's scheme allows either the customer or the vendor to communicate online with the bank for payment verification. Online secret sharing schemes are extensions to earlier protocols for authentication in distributed systems, such as Needham-Schroeder [NS78], Otway-Rees [OR87], and Kerberos [NT94]. The difference is that the payment order is now being authenticated instead of just the user.

3.3.1.2 Online Public Key

MSET [Mic98a], from the French electronic commerce consortium (eComm), now CyberComm, allows small payments to be integrated with the SET macropayment environment. The user authorises a payment to a specific vendor online, with an MSET broker. The broker aggregates the small purchases and later clears them as a single SET transaction. The online communication, along with the SET digital signatures which are generated by each party, makes MSET as heavyweight as many macropayment schemes. The advantage over SET is that the banking network is not contacted for every purchase.

NetBill [CTS95, ST95] was also designed to allow small payments for network delivered services. However, it is basically an on-line electronic cheque scheme (see Section 3.2.2) where both the user and vendor digitally sign and endorse the cheque. Each transaction requires eight messages but this is due to the provision of price negotiation and encrypted delivery. A large effort is also made to address fault tolerance and provide *atomicity of payments*, where the transaction either fully completes or appears never to have occurred. While we note that there has been recent work on the importance of fault tolerance in electronic payments [CHTY96, Tyg96, PW97, XYZZ99], we do not agree that it is important in a micropayment scheme if its presence significantly increases overheads. If, every once in a while, the value of a single micropayment is lost, it is not a huge concern. Every day, similar amounts are lost in malfunctioning vending machines or payphones and, more often than not, a lost penny is left lying in a crowded street.

Online schemes allowing small payments may be suitable for a few daily purchases or for use in a localised Intranet environment, but they will not scale for repeated payments such as paying for a voice call. It is

interesting to note that these schemes have been the ones commercially used on the Internet rather than other offline schemes. This is likely due to the broker's requirement to keep full control of the system and to maintain a per-transaction fee similar to the credit card model.

3.3.2 Public-Key Based Payments

Macropayment schemes require frequent use of digital signatures in every transaction for authentication of entities or payment instruments, as is evident from earlier sections. A number of small payment schemes have been proposed that reduce the number of signature operations in each purchase but do not completely eliminate them. These schemes are now examined.

NetBill and MSET, discussed in the previous section, both employ multiple digital signatures per transaction, but avoid the overhead of using a financial clearing network for every transaction. Beadle et al. [BGSB96] reduce the signature costs to a single signature generation by the user and a verification by the vendor in their offline electronic cheque-like scheme. Since the scheme is offline, there is no way to prevent the user overspending. In addition, since the vendor does not endorse the cheque, there is no non-repudiable proof that he received the payment, as with a traditional electronic cheque.

Mini-Pay [HY97], from IBM Research, limits the user overspending to a maximum pre-set value at each vendor. The user is issued with a daily spending certificate, a bank-signed upper spending limit for the user at any vendor that depends on their credit at the bank. The spending certificate is verified offline by each new vendor approached. Each Mini-Pay payment is a user-signed payment order, effectively an electronic cheque, which is verified locally by the vendor. If the user attempts to overspend at a particular vendor, an on-line verification is performed with the bank in order to obtain bank-signed permission to proceed. While infinite overspending is prevented, the user can spend up to the maximum amount at every vendor within the system. While software can deter cheating by the user, it cannot provide the real security of a tamper resistant device.

Finally, NetCents [PHS98] removes all possibility of user overspending, without collaboration with a fraudulent vendor. This is done by keeping user pre-paid value at a vendor, in the form of a bank signed temporary account in the user's name. When the user first visits a new vendor she instructs her bank to setup a new temporary account at the vendor. The user creates a public key pair to access the temporary account and sends the public key to the bank. The bank signs the account balance and public key together, before sending it to the vendor. The vendor now holds pre-paid value, authorised by the bank, which can be spent only by the entity holding the secret key corresponding to the account's public key. To make a purchase, the user signs a payment order, linking the new account balance to the vendor, with the account's secret key. This is an electronic cheque against the vendor-held account that can be verified offline by that vendor. The vendor claims actual value by later depositing these cheques with the bank. User overspending is not possible since the vendor will know how much remains in the account.

NetCents is interesting in that unspent account balances can be transferred from one vendor to another. This requires two digital signatures, one by the customer authorising the transfer, the other by the current vendor declaring the current account balance. Vendor fraud is possible in that the current vendor can sign a false account balance. While vendor-to-vendor transfer will reduce the online contact with a central bank, we are concerned about how likely competing vendors are to co-operate in transferring a user's spending power to a possible competitor.

These schemes do reduce a payment cost to as low as a single user signature and offline vendor verification. However, we believe that generating a public-key signature, or even verifying such a signature for every single payment, is still too computationally inefficient for repeated micropayments. In Section 4.2 we show

how public key algorithms, even using the most efficient implementations, are several orders of magnitude slower than symmetric cryptography. The use of public keys by each entity also implies the existence of a certificate authority hierarchy to certify, distribute and revoke public key certificates. We discuss the scalability problems of such an infrastructure for large global scenarios in Chapter 5.

3.3.2.1 Micropayment consolidation

Chomicki et al. [CNP98] explain how vendors can swap customer micropayment debts with each other in order to aggregate payments. Using this debt consolidation technique, a user's purchases at many vendors can be gathered into a large bill at a single vendor. The vendor then directly bills the customer using traditional billing. The scheme decouples the purchases from the actual payment.

As with other public key schemes, a payment is a signed purchase order, basically an electronic cheque. The signed purchase order represents money owed from a specific customer, the customer debt. A vendor will try and collect many debts from the same customer, so that he can bill that user for the total amount, rather than a single small amount. Customer debts are collected by swapping the purchase orders with other vendors. For example, a vendor might try to collect purchase orders for customer A and will swap a customer B debt with a different vendor to obtain another customer A debt. The original vendor, who is transferring the customer debt, signs a swap order which contains the customer purchase order. So, our first vendor would sign a swap order containing customer B debt and give it to the second vendor. In turn, the second vendor signs a swap order with customer A debt and gives it to the first vendor.

Vendors swap equal amounts of debt so that the net value transfer is actually zero. A swap order may be swapped again with another vendor, with another signature applied. This causes it to grow in size with each swap, in the same way that transferred electronic cash grows [CP92b]. When billing occurs, the user is presented with their original purchase order and any derived swap orders, and every signature is re-verified.

To aid debt swapping a global debt matrix, of size *(c * v)*, is used, where *c* is the number of customers in the system and *v* is the number of vendors. Each vendor maintains a column in the matrix, corresponding to their customer debts. A swapping server can be used to match possible swaps and debt matrix consolidation algorithms are specified in [CNP98]. For a large number of users and vendors the matrix may become unwieldy, with consolidation taking long periods of time.

A serious drawback of micropayment consolidation is the unlimited customer credit, which can only be halted by revoking the user's certificate. This is a consequence of replacing a central broker, who controls user credit and payment aggregation, with direct aggregation between distributed vendors. We note that debt swapping has similarities with the way in which the GSM toll tickets of roaming users are swapped. Both methods offer some desirable features not present in the other.

3.3.3 Hash Chain Schemes

It has been shown that many payment systems use a digital signature to authenticate the payment instrument to a vendor. In Section 3.3.2, we argued that a public key operation per payment is too inefficient for micropayments. Therefore, the problem is to find a method to authenticate each payment, but without using an expensive digital signature. One way to do this is to use shared secret keys, as described in Section 3.3.5. Other methods, using hash functions, are examined here.

3.3.3.1 Authentication using One-Way Functions and One-Time Signatures

A *one-way function* is a function that is easy to compute but computationally infeasible to invert. Evans et al. [EKW74] described how a one-way function can provide user authentication. The user computes the

outcome of a one way function, *y=f(x)*, and securely shares that outcome with the system to which it later wants to authenticate. The user is authenticated by revealing *x*, which could only have been known by her since *y* cannot be inverted due to the one-way property.

The authentication can be extended to messages to produce one-time signatures [DH76, Rab78]. A *one-time signature* allows a single message to be signed once using a set of private signing values and verified with a set of public validation values. Every binary bit in a message can be either 0 or 1. A one-way function can be used to authenticate each possible bit value, thereby requiring two function outcomes per message bit. Each function outcome is associated with one of the bit values, 0 or 1. A bit value is signed by releasing the secret value that maps to the function outcome assigned to that bit value.

Figure 3-2 shows how a 4-bit message can be signed using 8 different public function outcomes. As with user authentication, the outcomes are made public. To sign the message, only the secret values corresponding to the actual message bit values are released. To sign M=1001, the secret values x_7, x_4, x_2 and x_1 are released. Only one of the two secret *x* values per message bit is ever released. Since the secret *x* values are known only by the signer, the signature cannot be altered to sign a different valued message. One-time signatures, unlike alternative authentication schemes based on shared keys, have the advantage of allowing non-repudiation, or proof of authentication to third parties.

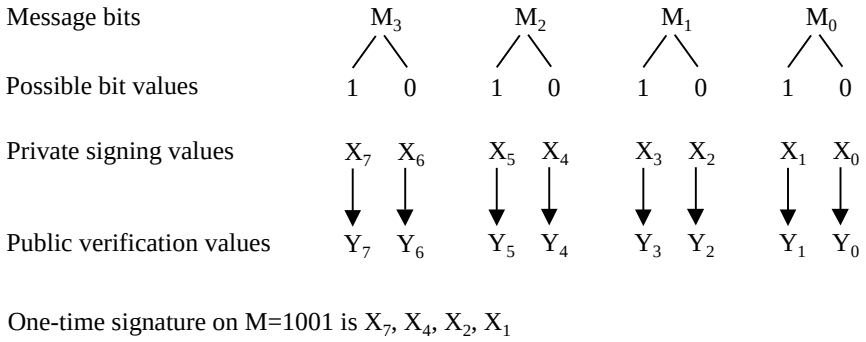


Figure 3-2 Simple One-Time Signature Scheme

Using efficent encoding techniques [Mer89, EGM89, Mer87], mentioned in Section 3.3.3.6, the number of one-way function outcomes required can be reduced. However, a problem is that the authentication can only be performed once, where ideally we would like many repeated authentications.

3.3.3.2 Hash Chains

Lamport [Lam81] proposed the use of repeated evaluations of a one-way function, to generate a chain of values, allowing many user authentications. A *hash function* is a one-way function that takes a variable length input, the *pre-image*, and produces a fixed length output, the *hash value*. A *collision-resistant* hash function is a hash function for which it is computationally difficult to find two different pre-images that map to the same hash value. For the purposes of this dissertation, the term hash function is used to refer to a collision-resistant one-way hash function unless otherwise stated. By using a hash function instead of just a one-way function, the collision resistant property prevents two different chains, or parts of a chain, with the same final value being found. A *hash chain* of length *n* is constructed by applying a hash function *n* times to a random value labelled x_n. The value x_n is called the *root value* of the hash chain. We define a hash chain derived using a hash function *h* recursively as:

$$h^n(y) = h(h^{n-1}(y))$$
$$h^0(y) = x_n$$

where $h^n(y)$ is the result of applying a hash function repeatedly *n* times to an original value *y*. The *final hash value*, or *anchor*, of the hash chain after applying the hash function *n* times is $x_0 = h^n(x_n)$. The hashes are numbered in increasing order from the chain anchor x_0, so that $h(x_1)=x_0$, and $h(x_2)=x_1$.

Each hash value in the chain can provide a single user authentication. The user releases x_1 for the first authentication, x_2 for the second and so on. The server only has to apply a single hash function to verify that the received value hashes to the previous value. The user only needs to store x_n from which the rest of the chain can be re-computed. As our measurements show in Section 4.2, hashing is highly efficient, approximately four orders of magnitude faster than generating a public key signature. Hash chain based user authentication has been implemented as the S/KEY [Hal94, Hal95, HMN+98] scheme. Winternitz proposed the use of hash chains for message authentication, as described by Merkle [Mer89]. Specially designed hash functions such as MD5 [Riv92] and SHA [NIST95] exist, but symmetric algorithms can also be used as hash functions [Mer87].

The final hash x_0 of a chain may need to be securely swapped across a network. A public key digital signature can be applied to x_0, to produce a *signed commitment* to the hash chain, showing it to be genuine. Such a user signature on the chain anchor is represented as $\{x_0\}Sig_{User}$.

Hash values from a user-generated hash chain can be used as authenticated payment tokens. The first micropayment schemes which independently proposed this idea were Pederson's phone ticks [Ped96], PayWord [RS96], NetCard [AMS96], and iKP micropayments [HSW96]. On the first payment to a new vendor, the user signs a commitment to that vendor with a new hash chain. By including the vendor identity in the commitment, the vendor is linked to the chain, preventing it being redeemed by other vendors. For each micropayment, the user releases the next payment hash, the pre-image of the current value, to the vendor. Since the hash function is one-way, only the user could have generated this value, and knowledge of it can constitute proof of payment. In essence, the hash chain links the correctness of the current payment to the validity of previous payments. Each hash value is worth the same amount, which can be specified in the commitment. A payment of *m* units is made by releasing the single hash which is the m^{th} pre-image of the current hash in the chain. The vendor only needs to apply *m* hashes to verify it.

A broker, or trusted third party, is introduced to aggregate micropayments to many different vendors. Actual monetary value is claimed by redeeming the highest spent hash token, along with the commitment, at a broker with whom the user has an account. The hash chain payment scheme is illustrated in Figure 3-3.

By using a hash chain, the computational cost of a payment is now a single hashing operation for the vendor, after the initial single digital signature verification for a new chain. Where a user spends *n* hashes from a chain to make *z* payments at the vendor the average cost per payment is *(n hashes + 1 signature)/z*. Thus, in the worst case, where a user only ever makes a single purchase from a vendor, the cost is similar to the public key schemes in Section 3.3.2. Therefore, as with the majority of micropayment systems, the scheme is optimised for repeated payments to the same vendor.

To address the scenario in which a user is going to make only a few purchases with a vendor, the iKP micropayments scheme directs these payments through a broker. The user pays the broker using a user-broker chain and the broker pays the vendor using a broker-vendor chain. An iKP macropayment [BGH+95] is used to pay for each chain individually. The broker vendor chain will be used for payments from many different users, making it more efficient than using many very short user-vendor chains. However, the

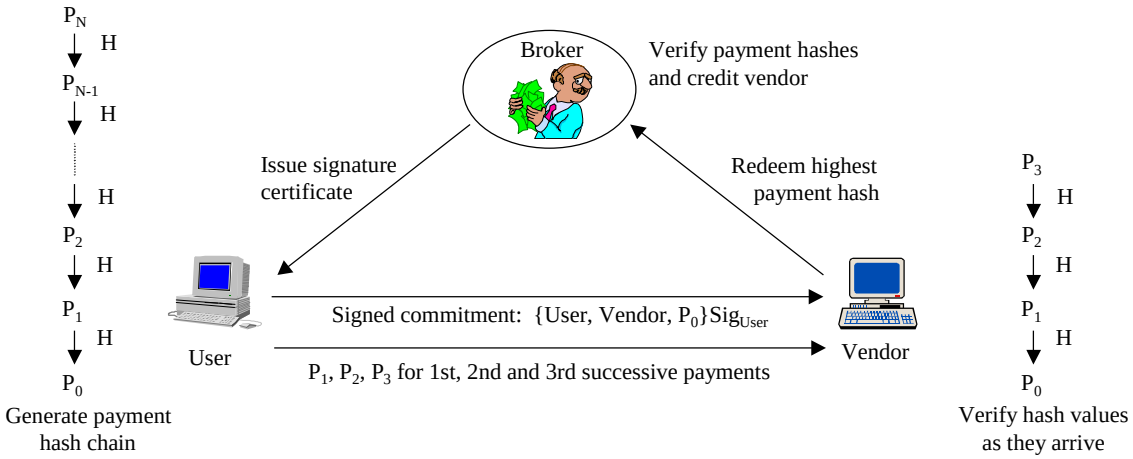


Figure 3-3 Hash Chain Payment Scheme

disadvantage is that there are two additional online connections by both the user and vendor to the broker for every purchase, making the communications cost greater than some of the online schemes in Section 3.3.1.

3.3.3.3 Applications of Hash Chain Micropayments

Use and application of hash chain micropayments in the literature and research projects is further evidence of their efficiency and practicality over other less cited alternatives. We illustrate the widespread potential of this technique, with the following examples.

Pederson's phone ticks were used to pay a single network operator for a phone call, as part of the ESPRIT project CAFE [BBC+94]. The ACTS ASPECT payment scheme, examined in Chapter 2, also used hash chains to pay a single VASP for Web services.

PDA-PayWord [DB99] implemented hash chain payments between the PalmPilot Personal Digital Assistant (PDA) and a physical vending machine. For digital signature generation on the commitment an elliptic curve algorithm was used instead of the slower RSA algorithm. However, on the server-side, RSA signatures were used as RSA signature verification, performed on the PDA, is faster than signature generation.

Franz and Jerichow [FJ98, FJW98] proposed hash chain payment for a mix-mediated message anonymity service, originally proposed by Chaum [Cha81] for anonymous e-mail transfer. Prepaid chains are generated on a smart card, which holds the bank's secret key, using a blind signature to provide anonymity. N-Count [KOO99] was designed to allow fast payment of road tolls from a user smart card. A universal shared smart card key allows the next hash value in a vendor-presented chain to be calculated. Compromise of the card breaks the security of these systems as discussed in Section 3.2.4.

Zhou and Lam [ZL99] designed a non-repudiable pay-per-view scheme [CO95] for a Web-based video service. It guarantees proof of viewing, using hash chains to allow fair billing. The lightweight security primitives for e-commerce proposed by Matias et al. [MMS97] include use of a user-generated but vendor-signed hash chain as proof of a user purchase. Hash chains are also proposed to prevent fraudulent billing in Web-based advertising by Reiter et al. [RAM98]. We draw attention to the similarities of the trusted log records of Web advertising, and CDRs of telecommunications networks, and note the problems of fraud in the former as emphasised by Reiter.

Non-payment applications using hash chains have also been designed. S/KEY [Hal94, Hal95, HMN+98] used hash chain values as passwords for user authentication. Hash chains have also been used for efficient

authentication of routing messages [HPT97, HPT99, Zha98], and multicast packets [Roh99]. In contrast to authentication, Hauser and Tsudik [HT96] employed hash functions to provide conditional anonymity for consumers during the pre-purchase browsing and service delivery phases of an e-commerce transaction. Micali's certificate revocation scheme [Mic96, Mic98b] linked a certificate authority (CA) hash chain to a user's certificate. A daily hash value from the CA indicates that the certificate has not been revoked, allowing efficiency improvements over certificate revocation lists (CRLs). Finally, Asokan et al. [ATW96, ATW97] used hash chains to provide non-repudiation of a user-generated message, signed by a trusted server.

3.3.3.4 Partially Spent Hash Chains

In a credit-based scheme, the user creates and signs a hash chain for a specific vendor. If the entire chain is not spent, the unused part can be safely discarded. No value is lost by the user since the vendor can only redeem the spent part of the chain. Where the user cannot be trusted with unlimited credit it is desirable to use a prepaid vendor-specific hash chain. This can still be generated by the user but is signed by the broker when it is bought. Since every hash in the chain represents prepaid value at a specific vendor, that value will be lost to the user if unused hashes are simply discarded. As with prepaid phone cards or prepaid gift vouchers, the entire amount must be spent with the designated vendor.

However, there have been schemes proposed which allow partially spent hash chains to be transferred for spending at another independent vendor. The basic idea is that the current vendor signs the highest spent hash in the chain, along with the chain details, to form *change*. When presented to a new vendor, the signature shows how much of the chain has already been spent, provided all previous signers of the change can be trusted.

In NetPay [DL99], unspent value is transferred by the current vendor signing the highest spent index in a user chain along with his identity and the identity of the new vendor:

$$\text{Change} = \{ID_{V1}, ID_{V2}, i\}Sig_{V1}$$

The chain transfer is passed directly between the vendors, upon request of the user at the new vendor. Unfortunately, this scheme is very open to abuse. A malicious entity can request a chain transfer on behalf of a user at any time, as no proof of user presence is required. Among competing vendors, this provides an easy denial-of-service attack. A user and vendor may collude to cheat other vendors and allow infinite double spending with post-fact detection, as in a credit scheme.

Vendor signed change is given directly to the user in Mao's lightweight micro-cash [Mao96]. The change consists of the last spent hash value, its index in the chain, and a user public key, all signed by the vendor:

$$\text{Change} = \{H_i, i, PK_U\}Sig_{V1}$$

The change, as with the initial chain, is not vendor specific. This allows unlimited user double spending as with credit schemes. However, the user public key in the change links the user to the partially spent chain. A user signature, computed with the corresponding private key, is required to spend the change at a new vendor. This allows post-fact detection of user double spending. User-vendor collusion is possible but with post fact detection of the vendor provided, all spent parts of the chain are later redeemed. Schnorr public key signatures [Sch89] are used, which can be largely pre-computed offline, and require only a hash function followed by a single modular multiplication and addition. However, Schnorr signature verification is not lightweight.

To provide user anonymity, a broker blind signature is used during purchase, and a new user public key pair is generated for each change request. The double spending detection scheme is such that use of the same private key twice reveals it, and hence there is a requirement for a new user key pair for each change instance. Thus, as with electronic cash, anonymity adds an additional computational overhead to the scheme. To spend new change requires 3 signature verifications by the vendor – on the original hash chain, the signed change and the user spending signature. Hash values can then be spent as normal with the vendor.

The change schemes remove a user communication with the broker for a chain refund or new chain purchase, and replace it with a user-vendor or vendor-vendor communication. The storage and computation costs are increased for both the vendor and broker. The vendor must verify and trust another vendor's signature, while the broker must keep track of those parts of a chain that have been redeemed and by which parties. The number of hashes performed is increased, as each party must verify that part of a chain belongs to the full chain.

The user must trust the vendor to issue change, as with the Millicent scheme discussed in Section 3.3.5. We also note that change relies on vendors co-operating with each other, which may not be in the interest of competing parties. Finally, we have shown how both prepaid schemes can allow unlimited overspending as with credit schemes. The cost of issuing hash chain change is the increased risk of fraud that accompanies the move from a model with no vendor trust to one requiring vendor trust with post-fact detection.

3.3.3.5 Hash Chain Trees and Graphs

A number of extensions and modifications to the basic hash chain scheme have been proposed. Some of these are as straightforward as adding vendor authentication as in iPay [Azb97]. However, others alter the Lamport chain structure to produce tree and graph structures, and we now examine these variations.

3.3.3.5.1 Authentication Trees

An *authentication tree* is a tree structure that allows multiple independent values to be authenticated through their connected branches to the authenticated root of the tree. In Merkle's authentication tree [Mer89], depicted in Figure 3-4, each leaf node is the value to authenticate. A parent node is a hash function of the concatenated value of its successors. The root node is initially authenticated by other means, such as by signing it with a regular asymmetric digital signature. To verify a leaf node the authentication path between the leaf and the root must be validated. Each parent node on the path must be re-constructed, requiring the value of each child node. For example to authenticate N_5 requires N_5, N_6, N_1, and the signed root N_0. An authentication tree need not be binary, and can be as deep as required. The authentication tree generalises the hash chain, since a hash chain is an authentication tree where each node has only one child.

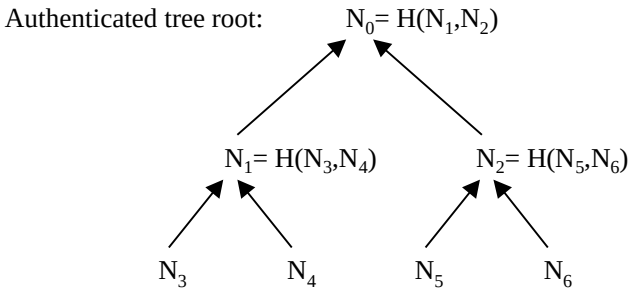


Figure 3-4 Merkle's Authentication Tree

Merkle first proposed authentication trees as a method of verifying multiple one-time signatures [Mer89]. A disadvantage of one-time signatures is that the public verification values, the set of one-way function

outcomes, must be securely authenticated by other means before a new signature can be accepted. Merkle constructed an authentication tree where each leaf node consisted of the verification set for a one-time signature. This forms a *k-time signature* scheme which allows *k* independent one-time signatures to be verified from a single set of public verification values. In Merkle's case, the one-time signature is verified as before, and its verification key is checked as belonging to the tree structure.

The original Merkle tree had a pre-determined size, limiting the number of authentications. In [Mer87], Merkle proposed an authentication tree that could be extended indefinitely. Each node consists of three one-way signatures, that is three verification keys. One signature is used to sign the value to be authenticated. The other two signatures are used to sign new right and left children nodes as required. This allows the tree to be extended indefinitely at the cost of greatly increased storage and the need to check all signatures on the authentication path.

Jutla and Yung based the PayTree [JY96] micropayment scheme on Merkle's first authentication tree. In PayTree each leaf of the authentication tree becomes the anchor of a normal hash chain, as illustrated in Figure 3-5. The root of the tree is signed with an asymmetric digital signature by a broker. Since the anchor of each hash chain can be authenticated through the tree, the need to sign each chain individually has been removed. As before, a different hash chain is used at each vendor. The contribution of PayTree is that, instead of a digital signature generation per vendor for a new hash chain commitment, the computational cost has been amortised to a single signature for *k* vendors. This makes PayTree more suitable for payments to many vendors, a characteristic of the MicroMint scheme in Section 3.3.4. This computational saving comes at the cost of increased storage and transmitted data, due to the need to verify the tree authentication path.

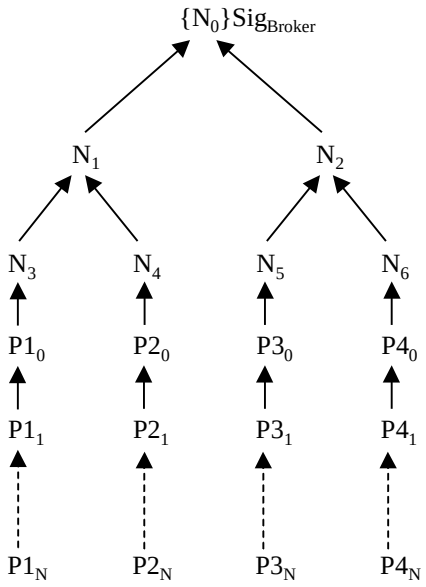


Figure 3-5 PayTree

User double spending of a PayTree is possible, with post-fact detection. Vendor collusion, to claim a chain spent at another vendor, is also possible unless the hash chains are made vendor specific as in a normal chain commitment. To address this, the public validation values of a one-time signature are placed on each node. The user, who holds the private signing values, can link a vendor identity to that node with the signature, thereby making the chain vendor specific. This increases the storage costs by the size of the one-time signature for every leaf. Assuming that only 5 digits are signed, allowing 10^5 vendors in the system, this will require 768 bits to be stored per one-time signature. This assumes an implementation using a 128-bit hash function, along with Winternitz and Merkle optimisations [Mer89]. For a tree with only 16 leaves, this is over 1,536 bytes of extra storage.

The tree structure of PayTree is used to link together multiple hash chains. It can be replaced by directly linking each chain together using a signature on the chain anchors:

$$\{h(P1_0, P2_0, P3_0, P4_0)\}Sig_{Broker}$$

This approach is used by Yen et al. to construct an Unbalanced One-way Binary Tree (UOBT) [YHH99] for micropayments. The disadvantage of this is that all the anchors of each chain must be sent to a vendor in order to verify the signature.

A drawback of PayTree is that the secret roots of each chain must be stored by the user. The UOBT solves this problem by deriving the root of each chain from another hash chain. In this way, multiple chains, which form the UOBT, can be generated from a single value, as shown in Figure 3-6. A hash function *h1* is used to derive the chain P_{ab}, P_{ab-a}, …. P_{2a}, P_a. A different hash function *h2* is used to derive a separate chain from each hash value in the original chain. A signature on a hash of the concatenated anchors links the derived chains together to form the UOBT.

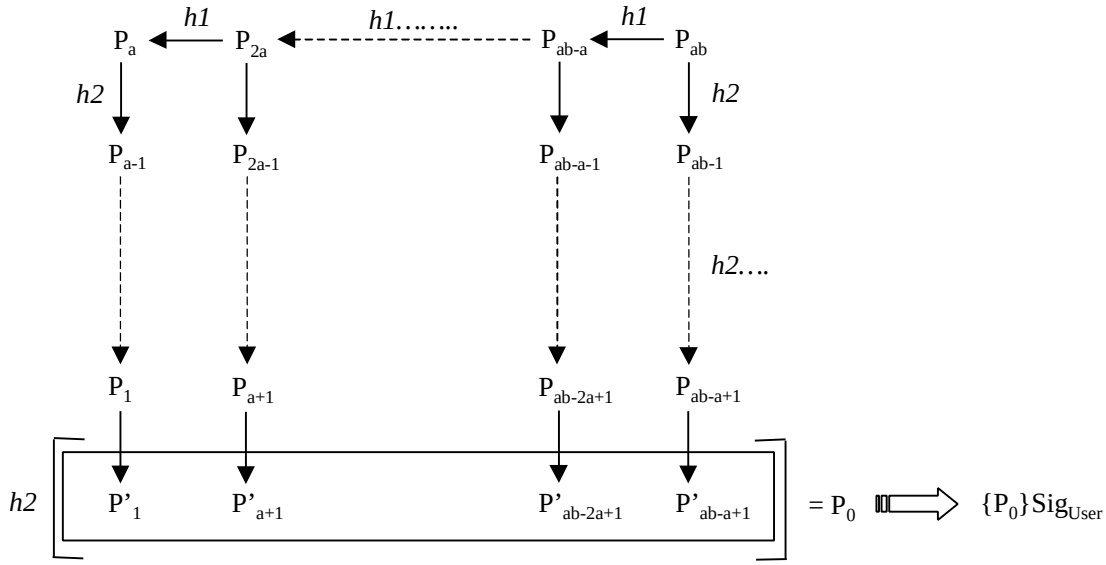


Figure 3-6 Unbalanced One-way Binary Tree

The UOBT was designed as an efficiency improvement over normal hash chain generation, where a single UOBT is spent entirely at a single vendor. On small devices with limited storage, such as a smart card, all values of a hash chain cannot be stored. A user must apply the hash function repeatedly to the chain root to obtain the next hash value to spend. On average, the number of hashes performed for a chain of length *n* is *(n-1)/2*. However, if a UOBT is used with $a=b=n^{½}$, the average number is $n^{½} - 1$. Thus the contribution is an efficiency improvement from O(n) to $O(n^{½})$. To further reduce the number of hashes, important parts of a hash chain or UOBT can be cached by the user.

Micropayment contribution #1: We apply the technique used to derive multiple chains from a single value to PayTree, as shown in Figure 3-7. The combined structure, which we call *UOBT-PayTree*, has the advantages of both schemes – the entire tree can be generated by the user from a single value and, because of the tree structure, each chain anchor is not required to authenticate a particular chain. To prevent vendors being able to derive other chains, the root of each chain is never released by the user. If a keyed hash function is used for *h1*, where the key is kept secret by the user, then the chain roots can safely be released.

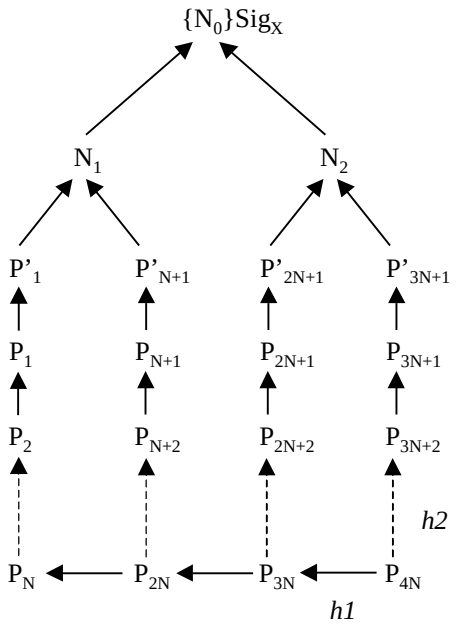


Figure 3-7 Combined UOBT and PayTree

Authentication trees for one-time signatures were generalised into directed acyclic graphs by Bleichenbacher and Maurer [BM94]. Miyano [Miy98] investigated the efficiency limits of one-time signatures on directed graphs. Domingo-Ferrer and Herrera-Joancomarti [FJ99] further generalised chains and trees based on hash functions into cyclic hash graphs, which they called spending programs. Their scheme allows hash values to be selected or re-used dynamically. This is implemented by using chain structures that can branch into two paths and later rejoin, or form loop paths back to an earlier part of the chain cycle. The idea was originally applied to guarantee program integrity [Fer90], but was later adapted for micropayments.

Each node in the hash graph consists of a hash of a parent node combined with a message, using an exclusive-or (XOR) function:

$$P_1 = h(P_2) \,?\, M_2$$

The message can be the monetary value of the node or a branch instruction, indicating which path to follow, when the node is connected to two child nodes. However, this requires that the message associated with each node be recorded when the graph is constructed, greatly increasing storage costs. While a value message allows each hash node to have different values, they must still be determined when the graph is constructed. As with a regular chain commitment, the final node in the hash graph is signed by the user, and the whole graph is used at a single vendor.

A forward conditional branch from P_2 to P_6 is illustrated in Figure 3-8. For simplicity, we do not show the message associated with each hash node P_X. Although not mentioned in [FJ99], we derive that a secret is needed for each branch to prevent a vendor who was given the short branch P_2, P_2', P_6 claiming the values in the longer branch P_2, P_3, P_4, P_5, P_6. This further increases storage costs.

A backward conditional branch, where parts of the graph are re-used several times, is more complicated. The authors suggest signing each branch and obtaining an on-line Kerberos ticket to prove that the user authorised

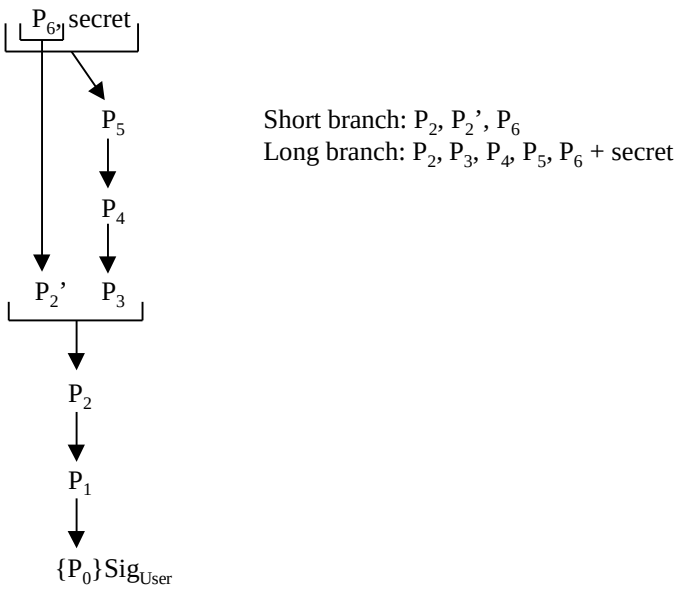


Figure 3-8 Forward Conditional Branch in a Hash Graph

the backwards loop. We suggest that the same level of security can be obtained off-line by releasing hash values from a separate hash chain to prove each re-iteration. This would require a hash chain to be associated with each possible backward branch. This has a similar overhead to generating a signature per branch if the branch is used only once, but is more efficient for repeated uses of the same branch.

A hash graph is only useful in a credit based scheme where a user can avoid additional digital signatures by using and re-using only parts of a payment graph. Tree and graph structures based on hash chains offer increased flexibility and improved efficiency in selected scenarios but at a greater storage and bandwidth cost than traditional chains.

3.3.3.6 Infinite Chains, Salting and Fault Tolerance

An alternative to reusing parts of a hash-based structure is to be able to extend it infinitely, as required. Pederson [Ped96] suggested the use of a trapdoor one-way function, such as RSA, for generating an unlimited payment chain. The chain can be extended by applying the user private key, SK_U, to the current root of the chain:

$$X_{n+1} = f^{-1}(X_n) = \{X_n\}SK_U$$

The matching public key is used to verify that the value belongs to the chain. However, due to the computational complexity of secure trapdoor one-way functions, they are not suitable for lightweight payments. Mu et al. [MVL97] propose an efficient hash chain extension method, called Unlimited PayWord (UPayWord). A chain is generated by hashing a root w_0 repeatedly with a sequence of secret values (s_x):

$$w_1 = h(w_0, s_0)$$
$$w_2 = h(w_1, s_1)$$
$$w_{n+1} = h(w_n, s_n)$$

The secret values are derivable from s_0 and are based on a shared user-broker secret. Unlike a normal chain, the user signs a commitment to the chain root and releases each following w_x as the payment token. Since the final hash w_n is never fixed, the chain can be extended indefinitely by continuing to generate further w_x values. However, because of the secret s_x, the vendor is unable to verify any of the tokens offline. He must

trust the user to send valid tokens or verify them online. Indeed, even if the user cheats, the vendor cannot later prove this.

Micropayment contribution #2: We suggest an alternative solution that allows the vendor to verify all tokens offline and that cannot later be denied by the user. We propose the use of a normal hash chain, but with the public verification values of an unopened one-time signature linked to the root of the chain, as shown in Figure 3-9. The one-time signature can be opened to sign the anchor, Q_0, of a new chain once the old chain has been exhausted. This is done by releasing the appropriate private signing values. In turn, another unopened one-time signature can be linked to that chain to sign further chains if required. Basically, we are replacing the digital signature on a new chain with a one-time signature.

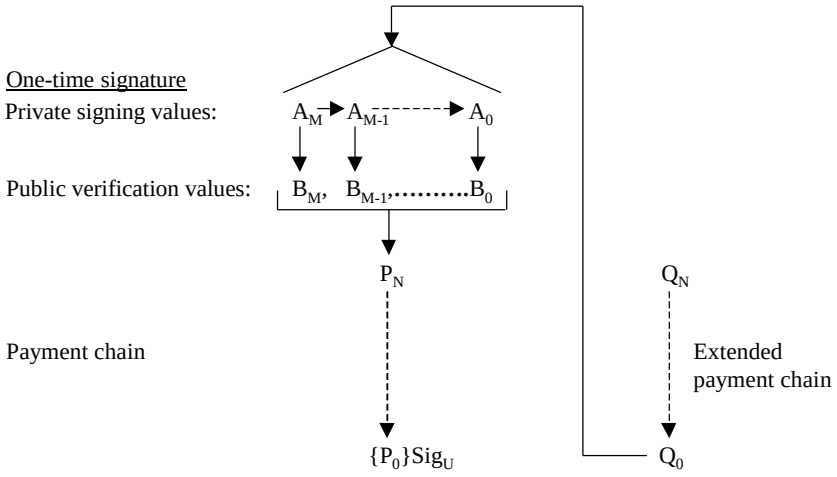


Figure 3-9 Infinite Hash Chain using One-Time Signature

A disadvantage is the size of the one time signature. If we use a 128-bit hash function, such as MD5, a Lamport signature would require (128 * 2) = 256 independent private signing values. These are the A_x values shown in Figure 3-9. Merkle's improvement reduces this to ($128 + \log_2 128$) = 135 values. Using hash chains to represent a group of bits, as proposed by Winternitz, we can represent x bits using a chain of length 2^x. Therefore, we could divide the 128-bit Q_0 message into thirty-two parts, each of 4 bits, which would require 32 hash chains of length 16. The selection of the number of chains to use is a speed/storage tradeoff.

A checksum is appended to the original message and signed, using another hash chain, to prevent alteration of a received signature. For example, a vendor might claim he only received X_4 when in fact he was given X_6, a hash higher up the chain. The checksum is a count of the unreleased parts of the chain, that is the number of hashes from the root to the highest released hash. Therefore, if the vendor cheats, by claiming a lower number was received, the corresponding checksum count must be increased. This will require higher hashes from the checksum chain, which cannot be obtained due to its unidirectionality. The checksum uses this inverse relation to protect the signature.

We observe that the presence of the checksum allows x bits to be represented with a chain of length $2^x - 1$, instead of the original Winternitz length of 2^x. This is because there is no situation where the message and checksum can both be zero. Even if the message is zero, the checksum will not be and vice-versa, due to the inverse relation. We use the chain root to represent the maximum number and the anchor to represent the minimum. Therefore, we use X_0 to represent 0000, X_1 for 0001, and X_{15}, the chain root, to be 1111 in a chain of length 15.

The maximum value of the checksum is (32 * 15) = 480 which can be represented in 9 bits, since 2^9=512. We need another 3 chains, two of length 15, and one of length 1 for the checksum. The total is 34 hash chains of length 15, and one chain of length 1 as shown in Figure 3-10. However, the user only needs to store a single value, from which these multiple chains can be derived using another hash function and chain, as with the UOBT.

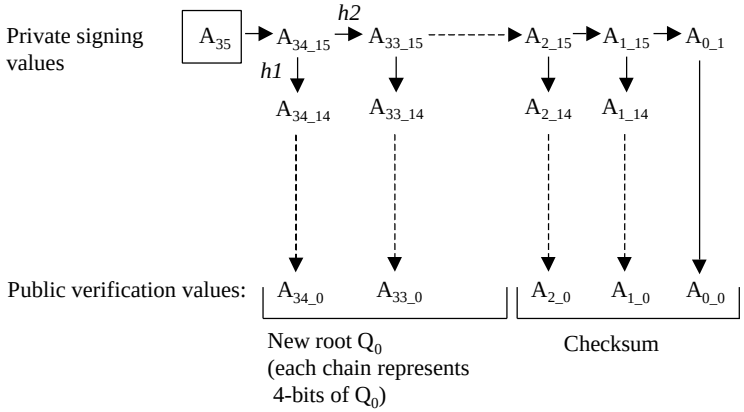


Figure 3-10 Optimised One-Time Signature Structure for Infinite Chain

Therefore, we require no extra storage for the user and 560 bytes (35 * 16 bytes) extra storage of the one-time signature for the vendor, if the chain is extended. The storage size of the one-time signature is approximately four times the size of a regular 128-byte (1024 bit) RSA signature, but less than three times the size of a new 60-byte commitment with the RSA signature. The number of extra hashes required by the user to generate the unopened signature is 545 hashes (34 * 15 hashes to generate the 34 chains of length 15, plus 1 hash to generate the chain of length 1, plus 34 hashes to generate the chain roots A_{M_15} to A_{0_15} from A_{35}). This number will be less for the vendor, depending on the message signed. To verify the one-time signature will require a minimum of 15 hashes (message all 0 bits, checksum 111100000 or 480), to a maximum of 480 hashes (message all 1 bits, checksum all 0 bits). This is still several orders of magnitude more efficient than a new RSA digital signature, as shown in Section 4.2. Using salting techniques, as described below, the hash function output size can be reduced, which would result in a smaller one-time signature and further efficiency savings. We now have a way of infinitely extending a hash chain that requires at most three times the storage of a new signed chain commitment, but is much more computationally efficient to generate.

Salting is a technique where the length of the input to a one-way function is increased by appending some random data. This serves to make a brute force or dictionary attack more difficult, as, due to the increased input length, more attempts must be made. Salting can be used with a hash function and hash chains to increase the security. When creating a hash chain, a salt is hashed with the original value as shown in Figure 3-11. Now, even if an attacker finds another value P_1' which hashes to P_0, it will probably not contain the user held salt S_1 and can therefore be detected as a forgery. A similar technique is used to detect forgeries in MicroMint, as discussed in Section 3.3.4.

Since salting makes it much more difficult to find a valid pre-image, the size of the hash function output can be reduced, perhaps to as low as 80 bits. This, in turn, will increase the hash computation speed, allowing all schemes which use the hash function to operate faster. Salting for hash chain micropayments was first suggested in PayWord [RS96], and Mu et al. [MVL97] proposed salted PayWord (SPayWord) where the set of salts was derived from a secret shared with the broker. This allows the broker to verify the salted hashes when they are redeemed.

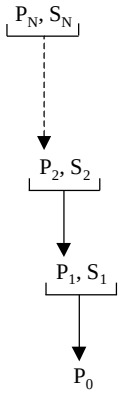


Figure 3-11 Salted Hash Chain

In the same paper [MVL97], Mu et al. propose Parallel PayWord (PPayWord) as a means to provide fault tolerance for hash chains. Each payment token is linked with the user commitment for verification if other tokens are lost in transit. However, we note that hash chains are inherently fault tolerant – if a hash, or even multiple consecutive hashes, are lost during transmission then they can be regained by repeated hashing of the next received valid hash. Fault tolerance is not a problem with hash chains, once the commitment hash been successfully received.

In summary, hash chains provide a highly efficient method of repeatedly authenticating a user or payment. However, the initial anchor of the chain must be authenticated using other means, usually an asymmetric digital signature. Each payment of *n* cents requires only *n* hashes to verify, if the minimum unit is one cent. Use of a customer signature allows unlimited credit, while the vendor must be known in advance for prepaid broker-signed chains. Vendor issued change for prepaid instruments increases the potential for fraud. Generalisations of the chain into trees and graphs spreads the cost of the signature over several vendors, but increases storage and communications costs. A digital signature on a new chain can be avoided by infinitely extending an existing chain. We proposed an efficient infinite chain based on one-time signatures.

3.3.4 Hash Collisions and Hash Sequences

A *hash function collision* occurs when two or more different pre-images are found that hash to the same value. Two distinct values x_1, x_2 form a *2-way hash function collision* if they both hash to the same value y_1:

$$h(x_1) = h(x_2) = y_1$$

Since a secure one-way hash function aims to be collision resistant, it is computationally hard to generate two pre-images that hash to the same value. This property can be used to define an electronic coin as a *k-way hash function collision,* as shown in Figure 3-12. The coin is created by an issuer who invests a large amount of computation to search for hash function collisions. It is efficient to verify that the coin is valid, by checking that each pre-image is unique and that they all hash to the same value. The collision is being used to authenticate the coin instead of a digital signature. The greater the length of the hash function output, the more difficult it is to find collisions.

Rivest and Shamir's MicroMint [RS97] was the first micropayment scheme to use this idea. Broker-issued payment tokens are defined as k-way hash function collisions, and no public key cryptography is used. Each token is worth the same minimum value, such as 1 cent, which can require the user storing and transmitting a large number of coins. Unlike other micropayment schemes, the tokens are vendor independent, in that they can be spent at any vendor.

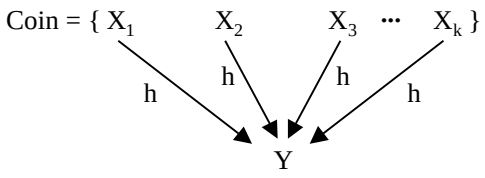


Figure 3-12 Electronic Coin as a k-way Hash Function Collision

Generating coins requires finding any k-way collision, which is a classic attack against hash functions called the birthday attack [Yuv79, MOV96, Sch96]. The *birthday attack* arises from the fact that with a large group of random people the probability that any two of them share the same birthday is far greater than the probability of randomly selecting one date and finding a person with the same birth-date in the group. For a group of only 23 people, the probability at least two of them share the same birthday is greater than 50%. The broker will perform a large number of computations, using computational resources unavailable to a normal adversary, to search for coins in advance. Finding the first collision will require many hashes, but the following collisions will require less work, allowing the broker to mint coins affordably.

Further protection against forgery is added by imposing a coin validity criterion, in which a portion of the hash value must equal a certain bit pattern. This has the added advantage of reducing the storage requirements of the bank, since non-conforming hashes can be discarded instead of storing them as candidates for a possible collision. Similar hidden requirements may also be required on the pre-images. In addition, the broker will record all collisions calculated by him, so that other new forged collisions will be detected. Finally, coins have a short validity period of a month, and the validity requirements are only released as the coins are, forcing an attacker to generate the coins during their short validity period. A *keyed hash function*, where a secret key is concatenated with the pre-image before hashing, can be used to further restrict pre-computation by an adversary. Further MicroMint implementation-specific security issues are examined in Burstein's thesis [Bur98] and in the Mimi [HBFH97] system.

While MicroMint tokens constitute a form of identified electronic cash, in order to be efficient there is no on-line verification with the broker to prevent double spending. If double spending occurs, the broker will know which user and vendor are involved, but cannot know which entity cheated. This scenario seems better suited to small localised solutions rather than global payment schemes.

The definition of a coin can be modified so that user-specific coins can be issued by the broker to prevent stolen coins being spent. The user identity *U* is hashed using a second hash function *h2* to produce a group of numbers, each labelled d_i. A coin is now a set of *k* pre-images $\{x_1, x_2,\ldots, x_k\}$ whose hash values $\{y_1, y_2, \ldots, y_k\}$ form a sequence where the difference between each hash value links them to a specific user identity U:

$$h2(U) = \{d_1, d_2, \ldots, d_{k-1}\}$$

$$y_{i+1} = y_i + d_i \pmod{2^U} \quad \text{for } i=1, 2, \ldots, k-1$$

Effectively, the hash values combine to form a hash of the user identity. Coin verification will require one additional hash computation per purchase.

Using a similar technique, the coin definition can be extended further to include not only user information but also vendor information. Unlike Millicent, discussed in Section 3.3.5, users do not have to purchase coins in advance for specific vendors. Rather, the user withdraws large blocks of successive coins, which can later be made vendor-specific by selecting the appropriate hash values linked to the vendor identity. Since the user is

given the responsibility for making the coin vendor-specific at the time of payment, this technique cannot prevent user double spending, but it will prevent the vendor re-spending coins it has received. In order to trace double spending MicroMint coins are only valid for one purchase, as with the majority of electronic cash protocols.

Wheeler proposed extensions [Whe96b] to MicroMint which reduce the storage and communications cost. His approach was to use a sequence of hash values, created by the broker, arranged in increasing order with a known set difference between the values. The user is given the group of pre-image values needed to produce the sequence of hash values. Each pre-image represents a single coin of unit value. Constraints are applied to the pre-images during coin creation to make forgery more difficult. The pre-computational costs are similar to that of MicroMint.

During a purchase, the user transfers the pre-images, or coins, corresponding to a specific range of unspent hash values, to the vendor. The number of coins per range is directly proportional to the range size, and each coin is worth the same amount. To reduce the communications cost, the vendor uses a challenge-response protocol to request only some of the pre-images of the hash values within the range. This has the added benefit of making it more difficult for the vendor to re-spend a range, since he will not know all the pre-image values to that range. The pre-images corresponding to the first and last values in the range are always sent. Since a coin is a single pre-image, instead of k pre-images in MicroMint, the storage costs are reduced by a factor of k. However, there is still a large amount of data to transfer and store compared to other schemes. Also, a minimum number of coins must be used per transaction to make forgery difficult. With a minimum range of 10 hash values, each coin will have to be worth one tenth of a cent to allow one-cent transactions, which in turn will increase the storage requirements. Thus there is little improvement over MicroMint for systems that allow transactions under 10 cents.

The security of these hash collision and hash sequence schemes rely on the premise that very large pre-computations performed by the broker cannot be performed by a determined adversary. This will require a large investment in specialised hardware, which will need to be regularly upgraded as cryptographic processors become faster and more widespread. Also, by generating coins in advance, a surplus of coins will have to be created to ensure that the maximum possible user requirements are met. Identified coins and hidden predicates are used to detect and limit large-scale fraud by an adversary with equal computational ability.

Recently, Jakobsson and Juels [JJ99] have proposed methods to outsource the bank collision computations to a large group of distributed untrusted devices, such as idle Internet-connected computers. Clients are given the task of finding hash values that match the coin validity criterion, which the bank then uses to find full collisions more quickly. The clients never see the secret key of the keyed hash function. Verifying a valid coin requires twice as many hashes as the original scheme, due to the blinding of the hash function key during minting.

The requirement that all coins have the same minimum denomination value will lead to a large number of coins being stored by the user. For example, with MicroMint, to store €20.00 where the minimum payment is €0.01, and coins consist of 4 pre-images (k=4), each of length 128 bits, requires a storage space of:

2000 * 4 * 128 = 1,024,000 bits = 1,024 Kbits = 128 KB

This is several orders of magnitude more storage than other micropayment schemes, and is prohibitively large for mobile devices with small storage capacity such as smart cards and palm computers.

The strengths of these schemes are that coins can be spent at any vendor and that coin verification uses only hashing. Many micropayment schemes require the user to know the vendor in advance when they purchase value from a broker. MicroMint coins are not linked to any vendor until the purchase transaction. Coins can be efficiently and independently verified. A MicroMint purchase of *v* cents requires *vk* hashes to verify, which is more efficient than a single digital signature (see Section 4.2). However, as with traditional electronic cash, double spending of the same coin at multiple vendors cannot be prevented but is detected after the fact.

3.3.5 Shared Secret Keys

A number of micropayment schemes eliminate the use of computationally expensive asymmetric cryptography and instead rely on *shared secret keys* between the parties. In Section 4.2, we show that symmetric cryptography is several orders of magnitude more computationally efficient than public key cryptography. Shared secret keys can be used to provide authentication and integrity by use of a message authentication code (MAC). A *keyed hash*, where the secret key *K* is appended to the message *M* and a hash function applied to the combined value, will act as a simple MAC. A more secure variant, HMAC [KBC97], is discussed in Appendix B. Symmetric encryption using the shared key will provide message secrecy. Secrecy can also be provided by using a one-time pad, instead of full encryption. To generate a *one-time pad* a random number *N* is chosen, hashed with the secret, and XOR'ed with the message to hide it:

H(N,K) ? M

N is also sent with the XOR'ed message, but the secret *K* is required to recover *M*. However, unlike public key digital signatures, symmetric cryptography cannot provide non-repudiation – that is, proof to an independent third party of message source – due to the shared nature of the keys.

While symmetric keys are more efficient, there is the problem of how initially to swap the secret value for a new relationship. Public key cryptography or out-of-band communications is used to solve this key distribution problem. In a large system, the number of shared secrets that each entity must securely keep can become unwieldy. In addition, each key needs to be refreshed periodically to prevent cryptanalysis and to limit the timeframe of a brute force attack.

3.3.5.1 Shared customer-vendor secrets

To remove the online communications cost with a broker, some schemes remove the use of such a broker altogether, keeping all communication just between the user and a vendor. Since one of the key functions of the broker is to perform payment aggregation between vendors, such schemes will require repeated regular purchases by a user, up to at least the value of a macropayment, at each vendor they use.

SubScrip [FW96] requires a macropayment to the vendor to set up a temporary user prepaid account. A shared secret, in the form of an account ID, is used to authenticate the user to the account. For each purchase the amount is deducted from the account value upon presentation of the account ID. Since the secret account ID is being sent in the clear across the network, a new secret account ID is issued after every purchase to prevent an eavesdropper spending the change. However, the new secret is also sent in the clear allowing it to be stolen. The authors suggest that, if this is a problem, the new secret can be encrypted with a user public key, $\{New\_secret\_accountID\}PK_{User}$. Unfortunately, a digital signature per purchase would eliminate the computational savings of using secret keys. SubScrip is therefore only practical in a network where eavesdropping micropayment value is not profitable.

Matias et al. [MMS97] propose the use of a persistent shared user-vendor key. This results in the user having to remember a large number of shared secrets, one for each vendor as with SubScrip. However, they allow the user to choose the secret, and derive it from a single user secret, K_C, combined with the user and vendor identities:

$$K_{CV1} = f(ID_C, ID_V, K_C)$$

Thus, the user only needs one secret, and all the derived shared secrets can be calculated on demand, allowing user device independence. A new secret is initially sent to the vendor, encrypted with his public key. The method does require that the same key is always used with the same vendor. Instead of refreshing the key, a derivative of it, such as $h(K_{CV1}, n)$ is proposed, where n is an integer that is increased each time a new key is needed.

The scheme only provides user authentication to the vendor, and the vendor uses traditional billing to obtain payment. Hash values from a user-generated, but vendor-signed, hash chain are used by the user to provide a purchase acknowledgement. While these show that the user received something, they do not guarantee that the price billed is accurate. Vendor cheating is possible in both this system and SubScrip.

3.3.5.2 Shared Entity-Broker Secrets

Millicent [Man95, GMA+95], which was one of the first micropayment systems to be proposed, uses a broker to aggregate user micropayments made to many vendors. There are vendor-broker and user-vendor shared secrets. A vendor issues value to users in the form of an authenticated message, called *scrip*, which specifies the value a user has at that specific vendor, rather like a temporary account. To ensure that scrip is genuine, the vendor could digitally sign it, but this is too computationally expensive. Therefore, the vendor uses a vendor secret in a keyed hash of the scrip to produce a MAC, as shown in Figure 3-13.

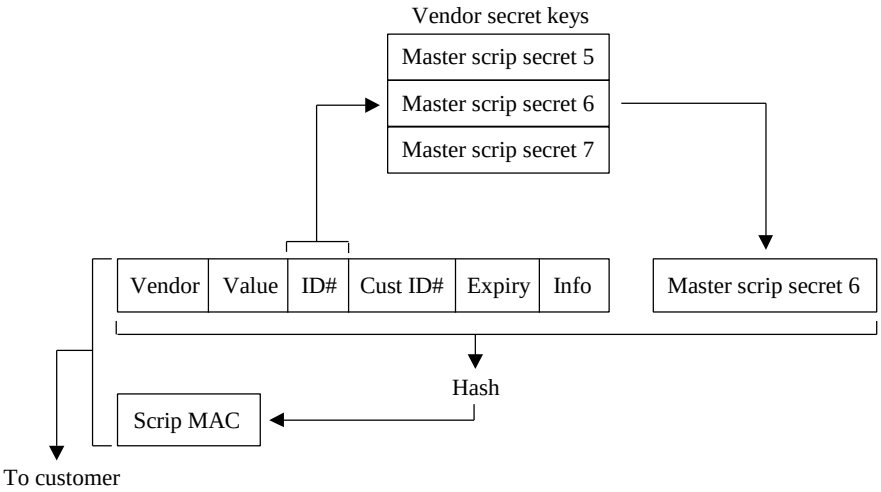


Figure 3-13 Generation of Scrip MAC

The MAC prevents the scrip being altered or forged. The vendor is, in effect, sending itself a message, the scrip which represents customer value, via the untrusted user. However, only the vendor can verify the scrip by re-computing the MAC. A serial number is placed in each scrip, as shown above, along with the value and other fields, to prevent double spending. Since scrip is vendor specific, it can be verified locally at the vendor during a purchase. A broker is used to sell vendor scrip to customers, avoiding a minimum macropayment to each vendor as in SubScrip. The vendor shares its MAC vendor secrets with the broker, in order to allow it to generate vendor scrip.

When the entire scrip value is not spent in a single purchase, the vendor issues new scrip with the remaining value as change. If the change is sent in the clear, it can be stolen and spent by eavesdroppers, as with the account ID in SubScrip. To prevent this, the customer and vendor share a secret, called the customer secret. This is given to the customer by the broker upon new vendor scrip purchase. The customer secret can protect the scrip and change using encryption. A more efficient solution, without the privacy, is to use it to generate a MAC on the scrip payment. The scrip change cannot be spent without knowledge of the shared secret.

The system is primed by making a macropayment to the broker to obtain value in the form of broker scrip. Vendor scrip is then purchased using this broker scrip. Since the value is vendor specific, the broker will be contacted online to buy vendor scrip for a newly encountered vendor. All further purchases with that scrip and its change are then offline. The key idea of Millicent is that value state is being held by the user, on behalf of a vendor, in the form of a secure vendor message. The user must trust both the vendor and broker not to cheat since she cannot independently verify the scrip message. Another drawback is the overhead of generating change and the possibility that the change is lost if the network fails. This loss would be more substantial than just losing a single micropayment.

Chrg-http [TL96] is an integration of the Kerberos authentication system [NT94] with the HTTP protocol [BFN96] to allow micropayments for Web resources. It is another example of a shared secret micropayment scheme where vendor cheating is possible. In Kerberos, both the user and vendor share a secret with a central Kerberos server. The user is authenticated online with the server and obtains a Kerberos ticket and a user-vendor session key K_{CV}. The ticket contains the user and vendor identities, and a copy of the session key, encrypted with the vendor-server secret (K_{VB}):

$$\{C, V, K_{CV}\}K_{VB}$$

The ticket is sent to the vendor, and a reply using K_{CV} provides mutual authentication of both parties. The vendor bills the user in the traditional fashion. While the ticket is proof of a transaction, there is no proof of the purchase amount, as with the scheme of Mathias et al. [MMS97]. Chrg-http suffers from the same scalability problems as Kerberos in a global environment. Every vendor must share a key with the user's local Kerberos server for cross-realm authentication. Chrg-http, along with the online secret sharing schemes in Section 3.3.1.1, show us that the authentication schemes of distributed systems are not suitable for micropayments outside an Intranet environment.

PayFair [YLH99] removes the possibility of vendor cheating that is present in the previous schemes. Millicent and other schemes are not *fair* in that vendor cheating cannot be proved to the broker. By combining hash chains and secret keys PayFair provides user fairness. Like Chrg-http, both the user and vendor share individual secrets with a third party broker. These secrets are used to generate keyed hashes (MACs) to protect all communication with the broker. Users buy a bank encrypted token, $(N)K_B$, where N is a unique serial number. Only the bank can verify this token, in the same way that, in Millicent, only vendors can verify vendor scrip. The user generates a hash chain using the token as the root of the chain.

To start payments to a vendor, the serial number N and final hash w_0 are sent to a vendor. Up to this point, the token has been vendor independent. The vendor makes an on-line connection to the bank where N, already associated with the user, is linked to the vendor identity. A positive response from the bank to (N, w_0) informs the vendor that the token is unspent and hashes can be accepted from the associated chain. The bank keeps a record of each token, (N, ID_U, ID_V), giving a similar storage overhead to the electronic cash scheme NetCash [MN93]. This storage cost is greater than other secret sharing schemes.

Basically, PayFair removes the digital signature on a hash chain, as used by schemes in Section 3.3.3, and uses a bank secret key instead. The drawback of this is that only the bank can verify a genuine hash chain and must be contacted online during the first purchase to do so. The hash chain provides the fairness in the scheme, while making it vendor specific on the first purchase prevents double spending.

Micropayment contribution #3: Noting the increased bank storage costs and online vendor communications of PayFair, we propose our own version of a secret sharing micropayment scheme with user fairness, which we call *PayFairer*. Users and vendors share a secret with the bank, to generate MACs as before. The user generates a hash chain, for use with a specific vendor, and gets the bank to commit to it. The user sends the final chain value w_0, and the length of the chain to the bank, protected by a MAC:

U -> B: ID_C, ID_V, w_0, len, h(ID_V, w_0, len, K_C)

The bank returns its secret key commitment to the chain details, $\{ID_V, w_0, len\}K_{SK}$, along with w_0 and *len*, encrypted with the vendor's shared secret:

B –> U: {w_0, len,{ID_V, w_0, len}K_{SK}}K_{VB}

The root of the chain, w_{len}, never leaves the user's device. The bank does not need to remember the chain details since that state is being off-loaded to the vendor. The vendor knows that the chain has been committed to by the bank because it is encrypted with his shared secret (K_{VB}). The user releases hash values in payment as with the regular hash chain schemes.

Our scheme removes the storage costs from the bank to the vendor. This is more scalable since there will be many vendors, over whom the storage cost is divided, while there is only one bank. We move the online communication of a vendor to the bank during the first purchase onto the user. Again, since there are many more users than vendors, this will prevent the vendor from having to make many online connections which would affect his server throughput. In addition, if the user knows which vendor it will use in advance, no online communication is necessary. Finally, the scheme strengthens customer fairness in that not even the bank can prove that a specific user chain is spent, until the user releases the appropriate hashes. Clearly, our scheme is similar to Kerberos but with the addition of a bank secret key commitment to a user generated hash chain.

3.3.5.3 Universal Shared Broker Key

A number of schemes rely on the use of a single bank secret (K_B) within tamper-resistant user or vendor devices. If the bank secret is ever discovered then the security of the system is broken and unlimited spending is possible without being distinguishable from genuine spending.

In Stern and Vaudenay's Small Value Payments (SVP) [SV97] scheme, the secret broker key (K_B) is present in all vendor's tamper-resistant devices. Users withdraw a token, which is certified with a keyed hash of K_B. A shared user-broker secret protects all communication between them. Since the token is verified with a broker key, it is similar to Millicent broker scrip. However, there is no value amount linked to the token as it is re-used as many times as required in the credit based scheme. The token is associated with the user at the broker to allow post-fact detection of user overspending.

Each time the token is spent, it is freshly authenticated to the vendor smart card using a challenge-response protocol. Knowledge of the token MAC (K_T), the K_B keyed hash, is required to construct a valid response.

The token MAC is never sent in the clear to the vendor, unlike Millicent, because it is used more than once. Instead it is used to generate a further MAC on the response message:

$$MAC(K_T, Challenge) \text{ where } K_T = MAC(K_B, Token)$$

K_T is the token MAC. The challenge consists of the vendor identity, and a smart card generated random number. The token fields and a user random number are also included. The vendor smart card can re-generate K_T using K_B and the token, and then verify the response MAC. Based on a valid challenge-response, essentially a fresh authentication of the token, the vendor smart card balance is incremented and a secure record of the transaction is logged. Later, the smart card balance and transaction logs are cleared, using MAC security, with the broker.

To limit the consequences of discovering K_B, the authors propose that each vendor device be given a subset of *n* broker keys. However, this then requires *n* different MACs for each token and *n* responses to the payment challenge, which decreases the efficiency of the scheme by a factor of *n*.

N-Count [KOO99], discussed in Section 3.3.3, uses a secret broker key on all user smart cards. The secret allows the card to generate any hash chain that the broker can generate. Thus the card can respond to a vendor request for the next hash value from a specific broker chain as payment.

As noted in [AK96, BDL97, BS97], tamper resistance may be overcome. For this reason it seems sensible to avoid large-scale deployment of schemes whose security relies wholly on it.

3.3.6 Probability Based Schemes

In the previous micropayment schemes, each and every payment is usually processed by the vendor and later verified and redeemed at a trusted third party. To minimise the number of transactions that must be performed, probability theory can be applied so that there is a specified likelihood or chance that the transaction will be performed. For example, instead of making 1000 micropayment transactions each worth 1 cent, one might make a €10.00 payment with a 1/1000 probability. Most of the time no payment will be made, but approximately every 1,000 transactions a €10.00 payment will occur giving an average cost of 1 cent. Over time, each party will get the correct amount on average. The micropayment cost has been eliminated for most transactions but the cost of fairly predicting a random event with known probability has been introduced.

In *probabilistic payment* schemes, there is a known probability, corresponding to the transaction amount, that the payment will actually be made. In *probabilistic verification* schemes, the payment is successfully validated with a certain probability. This includes *probabilistic polling* schemes, where the payee contacts a third party for payment verification with a certain probability, removing the need to verify every single transaction online. It also includes *probabilistic signature* schemes, where a signature verification can be computed more efficiently, at the risk of not detecting a forgery. We now examine micropayment schemes using each of these techniques in turn.

3.3.6.1 Probabilistic Payment

Wheeler proposed transactions using bets [Whe96a], where an *on-line coin flip* [Blu82] between the payer and payee is used to decide whether payment is actually made or not. A coin flip can be performed by the vendor choosing a random number *R* and committing to it using a one way hash function.

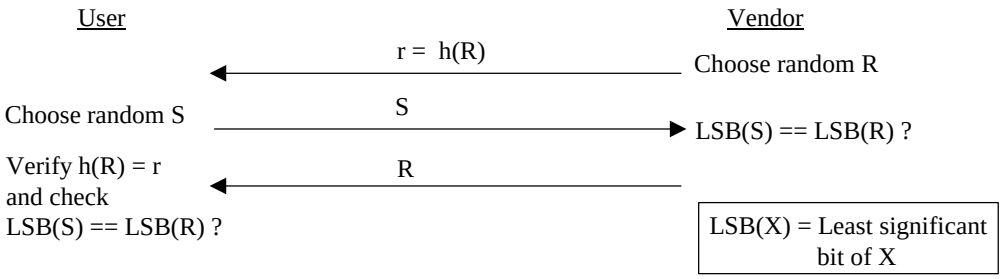


Figure 3-14 On-line Coin Flip

The hash value *r=H(R)* is sent to the user, who replies with a random guess *S*, as shown in Figure 3-14. The vendor then reveals *R* and the user verifies that it was committed to earlier by hashing it. The user wins the coin flip if the least significant bit of S equals the least significant bit of R, which will occur with 50% probability. For a single coin flip, only the least significant bits are compared. To generate multiple coin flips at once, a number of bits equal to the desired number of coin flips can be compared. A number of coin flip outcomes can be used to obtain a random number within a given range. For example, the outcome of four coin flips can be used to obtain a number between 0 and 15, by assigning a user win to be 1 and a user loss to be 0 in the appropriate four bit positions. This can be extended with the appropriate number of coin flips to randomly decide the outcome of any probability. For example, multiple coin flips can therefore be used to decide an outcome with probability 17/100, which would represent a 17 cent purchase price with a €1.00 coin. When required, actual payment can be made using one of the other micropayment protocols.

While the protocol requires fewer actual payments, the communication cost is increased as three messages are sent for each user-vendor transaction, rather than a single message. Another problem is that the vendor can increase his probability of success by aborting the protocol after he discovers an unfavourable outcome in step two. By forcing another new coin flip, the overall probabilities will be changed. Finally, there is no way to prove to a third party that the coin flip took place. A user could use this to deny that she owes payment to the vendor. While digital signatures could be used to sign the coin flip messages, a digital signature per transaction is too inefficient for a micropayment scheme. Lipton and Ostrovsky [LO98] overcome these two problems in their coin flipping micropayment protocol, described later in this section.

Rivest proposed the use of electronic lottery tickets as micropayments [Riv97]. The basic idea is to issue a user signed ticket containing a value which will be used to determine if it is a winning ticket:

$$\{\text{win\_indicator, win\_value}\}\text{Sig}_{\text{User}}$$

The ticket value will be the *win_value* multiplied by the probability of the ticket winning. Rivest suggests two types of indicators to decide a winning ticket. With an *external indicator*, the ticket *win_indicator* is compared with independently chosen numbers, such as those from a national lottery. The disadvantages here are that neither party knows the micropayment outcome until the external event occurs and the ticket must be held by the vendor until that time. With an *internal indicator*, the guesses from an online coin flip are included as the *win_indicator* (*r*, *S* in Figure 3-14). The vendor can immediately reveal the ticket outcome by revealing the blinded coin-flip value *R*. In essence, Rivest is applying a digital signature to step 2 of the coin flip so that the user cannot deny it.

To avoid using a digital signature for every transaction, the user and vendor can independently generate a hash chain each and include the anchors u_0 and v_0 in the signed ticket, as shown in Figure 3-15. As with

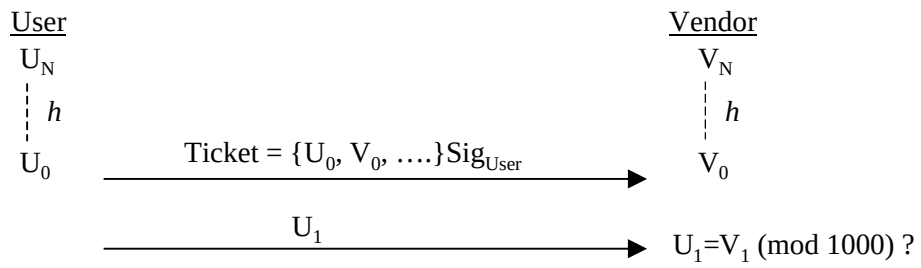


Figure 3-15 Probabilistic Payment using Hash Chain Lottery Tickets

PayWord, described in Section 3.3.3, a single hash value is released for each micropayment. However the ticket only wins if

$$u_x = v_x \pmod{win\_value}$$

Effectively, the corresponding hash values are acting as blinded coin flips, with the user revealing her coin guess as payment. While the computational cost is similar to PayWord, except for the cost of an additional hash chain, the bank only needs to process chains which belong to a winning ticket.

Yen and Ku [YK98] argue that the additional vendor chain adds undue extra computation requirements to the vendor. They envisage the vendor as having to perform *(n-x)* hashes to obtain v_x from the base v_n for each micropayment. This results in an extra *(n-1)/2* hashes on average per micropayment, in addition to verifying the user hash. However, in an efficient implementation, the chain, or certain key hash values in it, may be cached in memory to avoid such repeated hashing.

To reduce the vendor computation, they greatly reduce the length of the vendor chain so that it is much shorter than the corresponding user chain. Each different user hash u_x in each different payment is now compared to the same vendor hash v_y until a win occurs. After a win, the vendor uses the next pre-image v_{y+1} until the next win occurs. This saves the vendor performing unnecessary hashing, even in an efficient application.

Lipton and Ostrovsky [LO98] extend Rivest's lottery ticket scheme so that it is provably secure. To do this, both the user and the vendor must use a zero knowledge proof using cut-and-choose methods [BG92], to prove to the other that they know the base value of their own hash chain. The zero knowledge proofs, which are computationally more expensive than a digital signature, add an additional computational and communications cost for each signed ticket. This additional step is required to formally prove that the coin flipping protocol using hash chains is both fair and authenticated. This raises an interesting point, that to formally prove the security of a micropayment scheme may require adding cryptographic constructs that greatly decrease the efficiency of that scheme.

The protocol is *fair* and authenticated since the coin flip outcome is uniquely defined and can be proved to an outsider if the protocol is aborted. The scheme uses an online macropayment to make the actual payment to the vendor for a winning ticket.

Overall, the efficiency of probabilistic payments seems to be similar to other micropayment schemes for both the user and vendor. Due to the probabilities there will be variability in the amount paid, where an unlucky user will end up paying far more than expected. This gambling aspect may reduce user acceptance. The single advantage is that the bank needs to process fewer micropayments, only the winning ones, although the exact saving will depend on the implementation. This will also give greater anonymity, as the bank will not

see every transaction. However, the saving of some processing, already performed offline, is not a critical advantage in a micropayment scheme.

3.3.6.2 Probabilistic Polling

Schemes which use probabilistic polling are a hybrid between fully online and fully offline schemes. Many of the micropayment schemes examined, such as PayWord or MicroMint, give the user the opportunity to overspend their credit or double spend at *all* of the vendors. This could be prevented by verifying every payment online with the broker but this is too inefficient for small value transactions. Hence, to greatly reduce the maximum possible fraud while keeping the online communication small, a vendor can probabilistically choose whether to verify the payment online.

Agora [GS96] was the first scheme that used a *fixed probability* to decide if online verification should be performed. With probability *p*, or if the sum of user payments exceeds a specified amount, the bank is contacted and informed of the total spent at the current vendor. If a daily credit limit has not been exceeded, the transaction is allowed to proceed. Otherwise, the bank revokes the user's spending privileges and informs all vendors by broadcasting a *black list* of all revoked users. The probability that a vendor contacts the bank online is *p*, and the probability of no contact is *(1-p)*. Using this, we note that the probability that the first online verification is in the user's n^{th} purchase, *P(n)*, is a *geometric distribution*, where $P(n) = p(1-p)^{n-1}$. The mean of a geometric distribution is *1/p* [WM89, Mey70]. Therefore, a cheating user will be able to perform *1/p* purchases on average, after they have reached their credit limit, before being detected by the bank. The expected loss is therefore far less than would be the case were the user able to spend up to their full credit limit repeatedly at every vendor in the system.

Agora payments are digital signature based payment orders, which function as electronic cheques. With a vendor-signed price quote and the user-signed payment, we do not believe the scheme is efficient enough for repeated small payments, even though it was intended for use with transactions as low as 0.1 cent. It is possible for the user and vendor to collude and avoid online polling and, for this reason, the vendor is assigned liability if a user overspends with him. A further problem is the possibly large list of revoked users, the black list, that must be regularly updated at every vendor in the system. With a large number of users and vendors, user revocation lists do not scale well, and have the same problems as certificate revocation lists [NN00, MR00, Riv98, NN98, Koc98, FL98, Bra99, FL99, Mye98, Mic96].

Yacobi's Early Warning and Revocation (E-WAR) scheme [Yac97] is a smart card based offline electronic cash scheme that uses probabilistic polling of the cards transaction records, a probabilistic audit, to detect double spending. Smart cards that have been compromised are revoked and, as with Agora, the revocation list must be broadcast regularly to all vendors. The idea is to detect a compromised smart card before the attacker can double spend enough coins to recover the investment required to perform the attack. The probability of detecting fraud depends on the percentage of all transactions sampled, *d*, the number of coins double spent, and the number of times that each coin was spent. Yacobi shows that the probability of not detecting an attacker by the time he has regained his investment cost is $O(e^{(-cost * d)})$. The higher the cost of breaking the tamper resistance and the more transactions sampled, the higher the chance of detecting the fraud. Yacobi also shows that fraud detection probability is the smallest when each coin is double spent equally many times as other copied coins. Even though typically only 1% of transactions need to be sampled on-line, the use of digitally signed coins will prevent the scheme being efficient enough for repeated micropayments.

In the micropayment scheme of Jarecki and Odlyzko [JO97] the probability of a poll occurring is not fixed but varies with the amount the transaction is worth. This allows the bank to keep an accurate approximation

of the total amount spent. It also means that high value transactions will approach a fully on-line scheme while very low value transactions will only result in a small efficiency decrease over a fully offline scheme. Collusion is limited by only giving partial reimbursement to vendors who accepted payment from overspending users, although this is unfair to honest vendors.

The scheme removes the need for updated revocation lists to be kept at each vendor. Instead, the first transaction with a new vendor is always online with the bank. The bank thus knows each vendor the user has dealt with. For further transactions, the vendor probabilistically sends a poll message with the user's identity to the bank. Since the probability of this occurring is proportional to the payment amount, the bank uses the polling messages to estimate the total amount spent. If this estimated amount exceeds a certain threshold, all the listed vendors for that user are immediately notified and will reject further payments. The vendors then submit the actual payments to the bank and if, with small probability, the actual credit threshold has not been exceeded, the user may continue spending. Otherwise, the user's spending ability is revoked. The revocation list is kept at the bank and will be checked online by each new vendor. It is proposed that an existing micropayment scheme, such as PayWord, be used to make the actual payments. The polling rate is highly configurable but typically five to ten polling messages will be sent if a user spends up to their credit limit. Polling messages to the broker can be forged, as they are not authenticated. However, all this achieves is to add a delay while all the user's payments are cleared from each vendor.

Probabilistic polling schemes do strictly limit the maximum loss possible, which is more sensible than post-fact detection. The online contact during a purchase may be kept as low as 1%, depending on the risk to be assumed, which adds an acceptable communications overhead. However, we do not believe that the use of constantly updated user revocation lists will scale in a global scheme with large user and vendor populations.

3.3.6.3 Probabilistic Signatures

Lisiecki and Yerushalmi [LY95] designed a hash chain scheme where the user signature on the hash chain commitment can be verified more efficiently than a normal signature by the vendor. They used a modified Rabin signature [Rab78, Wil80] to allow the verification calculations to be performed with a much smaller range of numbers. A forged signature can be detected with a fixed probability, which depends on the amount of computation the vendor performs. The signature creation cost remains unchanged. However, signature verification cost is improved by an order of magnitude. Therefore, probabilistic signatures based on efficient public key algorithms can offer a significant performance improvement, but with increased risk of forged signatures.

3.4 Lessons Learnt

This chapter examined electronic payment systems, and in particular micropayments, highlighting the cryptographic techniques used and the communications requirements. An overview of macropayment systems was presented, focusing on the complicated public key based computations employed, especially with electronic cash. The overhead of these computations and the frequent online communications requirements of macropayments are quantified in the next chapter.

A thorough survey of the published micropayment literature was undertaken. Each scheme was grouped into one of six categories based on the cryptographic constructs used or the communications overhead. The groupings chosen were online small payments, public-key based, hash chain schemes and their derivatives, hash collisions and sequences, shared secret keys, and probability based schemes. As far as we are aware, this is the first detailed literature review of micropayment schemes.

The two cornerstones of micropayment research have been hash chains and shared secret keys. A series of small steps from these initial ideas provides continuing performance improvements. PayWord, and its derivatives PayTree, UOBT, and spending programs are an example of this incremental progression. Hash chains, and their derivatives, have received the most attention in the literature and are based on the idea of linking a single signature to a sequence of authenticated tokens, derivable from a single value. This makes them the most fault tolerant of all the schemes, as a previous payment token can be recovered from the next payment.

The majority of micropayments rely on user trust with post fact detection and blacklisting of fraudsters. There are only two options to verify a payment instrument offline without allowing double spending. The first is to use a trusted smart card to prevent the user cheating. The second is to use a payment instrument that is linked to a specific vendor and cannot be spent elsewhere. The schemes with user trust allow the user to make the payment instrument vendor specific. To remove this trust, the broker must perform this task, requiring an initial online connection if the vendor is not known in advance as with Millicent and prepaid hash chains. This is a tradeoff between security and communication.

3.4.1 Open Issues in Micropayment Research

There has been a substantial amount of research and proposals in the micropayments area, mainly from the cryptographic community, since the first papers appeared in 1995. As already mentioned, only a few of these have ever been implemented and there are almost no experimental results. Due to this, practical implementation issues and performance optimisations have not been considered. The next chapter, which estimates and analyses the performance of micropayment schemes, highlights some implementation decisions that significantly affect their performance.

Little work has been done to consider how micropayments can be integrated with existing signalling and service protocols. The selection of the appropriate protocol layer and placement of the tokens in control packets or user data will depend on the environment chosen. Possible protocols within which micropayments might be used include, but are not limited to, transport protocols such as TCP, UDP, IP, GSM, WAP, HTTP, SS#7, quality of service (QoS) reservation protocols such as RSVP [BZBH+97, Whi97], BGRP [PHS00], YESSIR [PS99], SRP [AFB98], and negotiation protocols such as RNAP [WS00].

In fact, micropayment research has been mainly confined to the Internet scenario of paying for information goods from a Web server over HTTP. User payments to a single vendor has been the only environment considered. As envisioned in Chapter 1, we see a tremendous potential for *multi-party micropayments*, where a user pays multiple parties at the same time, not just for information goods, but for transport services, quality-of-service, and bandwidth reservations. This is a whole new problem with its own variations. One such consideration is whether all the parties involved are being paid at the same time, a *synchronous multi-party micropayment*, or whether a sub-group of one or more receive payment at one instance while the others do not, an *asynchronous multi-party micropayment.*

User-to-user payments are also important and, while they may be possible with some of the surveyed schemes, they have not been investigated. There will also be new issues to deal with when introducing micropayments into a mobile environment. For example, it will be necessary to handle ongoing payments during handover between cells, particularly if those cells are independently administered by two different network operators.

A single broker model has only been considered in all of the presented schemes. A real-world application will require hundreds of distributed independent brokers. This, in turn, introduces new problems, which can

ultimately affect the contents of the micropayment token. The security and protocol for token withdrawal and redemption also needs to be addressed. The timely distribution, presentation, and verification of both static and dynamic pricing of services and information, payable with micropayments must be defined. The World Wide Web Consortium (W3C) has produced a way to embed static pricing for per-fee hyperlinks in a Web page [WWW99]. However, the problem becomes more complex when considering more dynamic services such as network access, packet transport, and quality-of-service.

Szabo [Sza96] has identified the need to provide simple user interfaces to micropayment systems. The effort expended by the user should not be worth more than the actual micropayment value, suggesting the use of profiles and agents to automatically make a decision transparently on the user's behalf. The design and implementation of multi-party schemes, and consideration of these open issues with those multi-party schemes, is presented in Chapters 5, 6, and 7.

Finally, it is possible that further improvements or new paradigms for micropayment protocols, such as the idea of using hash collisions, may yet emerge. Our own survey of existing micropayment techniques yielded three new contributions. We proposed an efficient verifiable way of infinitely extending hash chains, proposed a shared key scheme that removed vendor trust, and showed how to derive and store hash chain trees more efficiently.

In the next chapter, we quantify the performance of the cryptographic constructs underlying all electronic payments. The results are then used to directly compare the performance of micropayment schemes from each of our six categories. This comparison, along with the lessons learnt from the micropayment survey, is then used to decide which techniques are best-suited for our own multi-party micropayment scheme.

4 Micropayment Performance

"Performance stands out like a ton of diamonds. Nonperformance can always be explained away."
Harold Geneen, former Chairman, International Telephone and Telegraph.

4.1 Motivation

A detailed survey of micropayment schemes was presented in Chapter 3. It explained the cryptographic techniques and message exchanges that take place in each different system, and categorised the schemes based on these. However, since only a handful of the schemes have even been prototyped, there is no real understanding of how they perform against each other in terms of storage, communication, and computation. This chapter estimates and compares the performance of twenty representative micropayment schemes, spread over the six categories defined in the survey. By analysing the performance characteristics, and knowing the cryptographic techniques employed, appropriate constructs can be chosen for use in our multi-party micropayment scheme in Chapter 5. The importance of adequately estimating performance and scalability requirements in e-commerce systems has also been noted by Murphy [Mur00].

In Section 4.2, the computational performance of cryptographic algorithms, upon which payment systems are built, are examined. The measurements concentrate on a Java implementation of the algorithms, benchmarked using a commodity personal computer. However, in Section 4.2.4, figures are also presented for implementations in different programming languages and on different hardware platforms, including mobile devices. A discussion of possible alternatives for slow public-key algorithms appears in Section 4.2.5.

Since Chapter 3 explained which cryptographic constructs were used in each system, the cryptographic measurements can be used to estimate the performance of each micropayment scheme. The space occupied by key material and payment value is compared in Section 4.3.1. The size and frequency of communications traffic is presented in Section 4.3.2. The computational performance, based on the cryptographic measurements, is compared in Section 4.3.3. In order to verify the unsuitability of macropayments for frequent low-valued payments, the performance of a collection of representative macropayments is estimated and compared to micropayments in Section 4.4. Finally, in Section 4.5, conclusions are drawn based on our performance analysis.

4.2 Performance Comparison of Cryptographic Algorithms

In order for micropayments to be efficient and scalable, they need to maximise the use of lightweight cryptography. It is assumed that public key cryptography is much more compute intensive than traditional symmetric cryptography, which, in turn, is considered less efficient than hash functions. For example, the RSA algorithm relies on exponentiation of large numbers, which requires many repeated multiplication operations on a standard CPU. However, the micropayment literature does not adequately compare the computational costs of existing algorithms, but bases their design on rough assumptions. In 1996 Rivest estimated that a typical workstation could generate 2 RSA signatures per second, verify 200 RSA signatures

per second, and compute 20,000 hashes per second [RS96]. The following year, the Millicent creators estimated that 20 RSA signatures, 100 RSA verifications, 10,000 DES encryptions, and 100,000 hashes, per second were possible on a commodity computer.

The purpose of this section is to use concrete experimental measurements to accurately compare the performance of specific cryptographic algorithms on typical hardware platforms. Specifically, we have the following objectives. The number of operations that can be performed per second by each cryptographic algorithm needs to be found. We need to be able to compare the computational efficiency of one algorithm against another. The feasibility of using a Java implementation, over more native Assembler or C++ implementations needs to be investigated. This will allow us to assess its suitability for use in our multi-party payment prototype. In addition we need to see if different implementations of the Java language offer significant execution speed differences with cryptography. Such differences would directly impact on our prototype measurements due to frequent use of cryptography in our payment solution. In addition, payment messages are typically small in size and the effects of key setup and padding overheads on these messages needs to be established. The speed comparisons of cryptographic algorithms executing on small messages is likely to be very different than those on traditional large messages, due to these overheads.

Our roaming scenario in Chapter 1 allows users to connect to a network using many different access devices. The processing power of a typical PDA, mobile phone, smart card, and handheld PC is considerably less than a workstation. The number of cryptographic operations that can be performed on each device needs to be investigated to quantify the limitations of these mobiles. Finally, we need to investigate if there are any more practical or efficient asymmetric digital signature alternatives to RSA. The results from these different studies allow us to proceed in Section 4.3 to directly compare the performance of the surveyed micropayment schemes, and are used in the design of our own multi-party micropayment system. Assumptions about cryptographic speeds are also made, but not verified, by many other proposed systems. For example, the VersaKey group key management system replaces an RSA signature with hundreds of encryption functions to improve efficiency [WCSW+99]. Our performance measurements will also allow such claims to be independently validated.

A number of widely used cryptographic algorithms which are considered secure were selected. MD5 and SHA1 were chosen as the hash functions, DES, Triple DES (3DES), and RC6 [RRSY98] as the symmetric ciphers, and RSA as the asymmetric algorithm used for public-key digital signatures. RC6 is a block cipher designed by RSA Laboratories to meet the requirements of the Advanced Encryption Standard (AES), the forthcoming encryption algorithm which will replace the use of DES by the U.S. government. We chose RC6 over other AES candidates because it has been shown to be faster for encryption, on a Pentium CPU, than any of the other AES candidates [SKWW+99, SW00, Dra00, SL00].

We used J/Crypto v.3.0 [Bal98a] as a pure Java implementation of these algorithms, except for RC6 where we used the optimised Java code which forms part of the RC6 AES submission. Crypto++ v.3.1 [Dai99] was used as a C++ implementation for all the algorithms. Our measurements were obtained using a 400MHz Pentium II microprocessor with 196MB RAM, running the Windows NT 4.0 Service Pack 3 operating system. To allow for the effects of process swapping and use of random data, each measurement run was repeated three times and the average calculated. A summary of our benchmark methodology and the resulting measurement data can be found in Appendix A.

4.2.1 Relative Performance of Cryptographic Algorithms in Java

To obtain our initial measurements, we used a Java implementation of the cryptographic algorithms on the Sun Java Virtual Machine (JVM). We chose the Java language [GJS96] due to its portability, popularity, and

simplicity, and because Java will be present on both mobile and fixed devices. This makes it suitable for a widely deployed payment solution, as outlined in Chapter 1. While JVMs initially yielded much slower program executions than C/C++, incorporation of Just-In-Time (JIT) compilers [CFMS97] and mixed mode compilers, have increased speeds substantially. A JIT compiler dynamically translates the Java bytecode into native instructions before executing them, while a mixed mode compiler adaptively selects which parts of the code to translate into native code.

Figure 4-1 shows the number of 64-byte blocks of input data that can be processed per second by each of our chosen algorithms, in Java. We refer to this as the number of operations per second that each algorithm is capable of, and our measurement methodology is described in Appendix A. We can perform over 115,000 MD5 hashes, 53,000 RC6 encryptions, 23,000 DES encryptions, 220 RSA signature verifications and 18 RSA signatures, per second. A logarithmic scale is required to illustrate such huge differences. We observe the difference in hash function speeds, where MD5 is twice as fast as SHA-1. However, SHA-1 is considered the more secure of the two [Dob96], due to collisions having been found for the MD5 compression function.

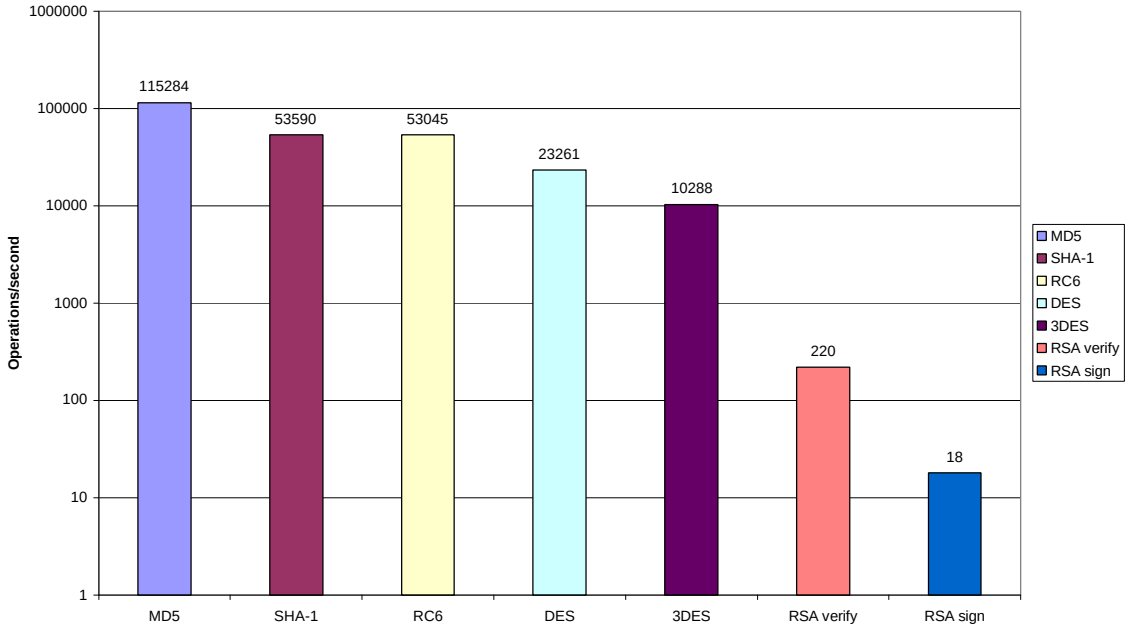


Figure 4-1 Efficiency Comparison of Cryptographic Functions in Java

We note that RC6 and SHA can perform an equal number of operations. This shows that the conventional belief that hashing can be performed much faster than encryption no longer holds true for some algorithms. A number of the micropayment schemes examined based their design on this assumption. Our C++ measurements, shown in Figure 4-5, find RC6 to be 22% slower than SHA.

A symmetric cipher can be used as a one-way hash function. One popular method [ISO94] to do this is to encrypt the message block M_i, using a function of the previous encrypted block M_{i-1} as the key K:

$$H_i(M_i) = E_K(M_i) \oplus M_i \qquad \text{where } K = H_{i-1}(M_{i-1})$$

The size of the hash result is equal to the cipher block length. Thus, for RC6 we would have a 128 bit hash, but DES would yield only a 64 bit hash, which may be too short to prevent collision attacks in many applications. Preneel [Pre98] discusses other slower methods where the hash is at least twice the block length.

Our measurements suggest that RC6 is as efficient as SHA and could be used as both the encryption and hash function in an implementation. Our graph shows speeds for processing large data blocks with the same key, where the key setup overhead is therefore negligible. However, from the above equation, we see that hash functions based on block ciphers require a key change after every encryption. This will impose a key setup overhead which is examined in Section 4.2.3. By using large input blocks, significantly larger than the block size of each algorithm, time is not spent padding small input data. The effect of padding small messages is also investigated in Section 4.2.3.

A surprising observation is that 3DES is only approximately twice as slow as DES, even though 3DES consists of a DES encryption, followed by a DES decryption, and a final DES encryption [NIST99]. A similar ratio was found in Preneel's measurements [PRA98], but not explained. This speed ratio remained even when the JIT compiler was removed, indicating that it was due to an optimisation in the J/Crypto code. Some implementations that we examined, including Crypto++ and Cryptix [Sys99] simply invoke their DES code three times, which forces 3DES to be exactly three times slower than DES. However, an optimisation can be made after closer examination of the DES algorithm. DES consists of an initial permutation, followed by 16 rounds of identical operations, followed by a final permutation, which is the inverse of the initial permutation. In 3DES the final permutation of the first DES encryption is cancelled out by the initial permutation of the second DES step. Similarly, the permutations can be removed between the second and third DES steps. Thus four permutation operations can be removed. Since the bit-wise permutation is relatively slow in software, this optimisation results in the substantial speed saving that we observed. Theoretical cryptographic speeds can be significantly improved by implementation optimisations.

Overall, hashing can be up to an order of magnitude faster than symmetric encryption, three orders of magnitude faster than signature verification and four orders of magnitude faster than signature generation. To

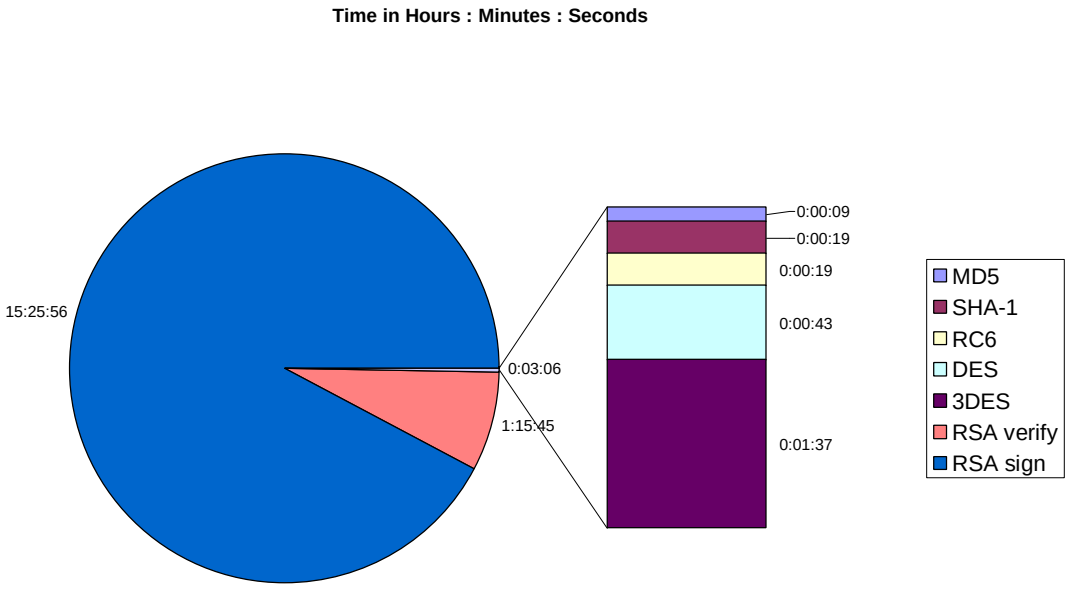


Figure 4-2 Time to Perform One Million Operations in Java

put this in perspective, consider the time it would take to perform one million operations of each, as illustrated in Figure 4-2. It would take over 15 hours to generate the signatures compared to 9 seconds to generate the hashes.

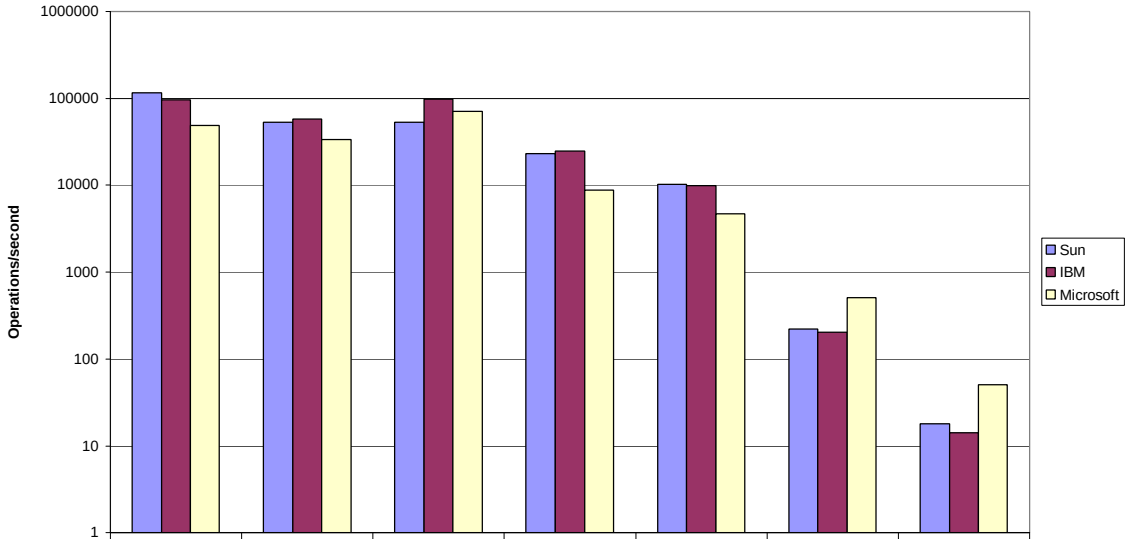
Further speed improvements are possible. We found that the Sun implementation of MD5 and SHA ran twice as fast as the J/Crypto implementation, illustrating the value of highly optimising Java code where speed is crucial. A compiler that directly translates Java source code into native code will yield further speed improvements, at the cost of removing portability.

To check that these performance differences were not just specific to Java, we performed the same measurements with a C++ implementation. While considerably more operations were possible, the same order of magnitude differences existed between the algorithms, as shown in Figure 4-5. Clearly, micropayment schemes should minimise RSA signatures and exploit the speed of hashing.

4.2.2 Cryptographic Performance on Different JVMs

JVM benchmark programs such as jBYTEMark [GR98] and VolanoMark [Nef99] show us that JVMs from different companies have different strengths and weaknesses. We investigated if there was any significant cryptographic performance change using three different JVMs with the same J/Crypto implementation. We chose the Java 1.1 JVMs from Sun Microsystems, Microsoft, and IBM. Both jBYTEMark and VolanoMark scored the IBM JVM significantly higher than the other two for all traditional benchmark operations, as shown in Appendix A.

However, this was not the case with cryptography, as shown in Figure 4-3. The Microsoft JVM was significantly worse at hashing and encryption, but substantially improved RSA signature throughput. It signed 50 RSA messages in the same time that the Sun and IBM JVMs signed 18 and 14 messages respectively. The Microsoft JVM also was able to verify twice as many signatures as the other two.



	MD5	SHA-1	RC6	DES	3DES	RSA verify	RSA sign
Sun	115284	53590	53045	23261	10288	220	18
IBM	96994	58356	97600	24738	9770	203	14
Microsoft	49199	33407	71467	8762	4666	508	50

Figure 4-3 Java Cryptography Performance on Three Different JVMs

On both the IBM and Microsoft JVMs we see that RC6 is the fastest algorithm, allowing more operations than either of the hash algorithms. However, the fastest algorithm on any JVM is MD5 on the Sun JVM. This is 15% faster than the next fastest which is RC6 on the IBM JVM.

These interesting results shows us that implementation tools should be carefully benchmarked for payment applications. The JVM chosen will affect application performance measurements and maximum load. Where signatures are used extensively, such as in a macropayment system, the Microsoft JVM is more suitable, while if only hashing and symmetric cryptography is used, such as in Millicent, the other two JVMs are more appropriate.

4.2.3 Hash functions vs. Symmetric Algorithms for Small Messages

Our earlier speed measurements were based on processing large 50KB chunks of data. With symmetric ciphers, time must be spent transferring the secret key into the form used by the algorithm. For example, in DES, the 56 bit key is transformed into sixteen different 48 bit subkeys, and each of these is used as input in one of the sixteen rounds of DES. When a large block of data is encrypted with the same key, the key transformation only needs to be performed once for the entire data and therefore does not delay the overall encryption speed. However, if a short input block is being encrypted, the key setup time will be a more significant percentage of the overall encryption time. For short input messages, encryption performance depends on both the key setup speed and the encryption speed. Since hash functions do not depend on a key, we would expect the performance of ciphers to decrease compared to hashing for small input messages.

On the other hand, the cipher algorithms process data in much smaller blocks than the hash algorithms. DES and 3DES use 64 bit blocks, while RC6 uses a 128 bit block size. In contrast MD5 and SHA use a 512 bit block size. Small input messages are padded with redundant data up to the required block size. The previous graphs showed the number of operations per second on a 512 bit message. If this message size is reduced, we would expect the performance of the ciphers to increase, but the hashing performance to remain constant.

Typically, in a micropayment scheme the messages are small. To examine how the algorithm's performance would be affected for small messages, we measured the time taken to perform the algorithm function a fixed number of times on small messages ranging from 1 byte to 264 bytes. From this, the number of operations per second for each message size was calculated and is shown in Figure 4-4.

The graph shows that the hashing algorithms are significantly more efficient than the encryption algorithms for all small message sizes. MD5 is three times as fast as the fastest cipher RC6, while SHA is approximately 1.5 times as fast as RC6. With messages up to 55 bytes over 93,000 MD5 and 52,000 SHA operations can be performed per second. In contrast, only 21,000 RC6, 16,500 DES, and 7,000 3DES operations can be performed on a 55 byte message. This indicates that traditional hashing algorithms are the most suitable cryptographic primitives, in terms of efficiency, for micropayment solutions. Optimal results are obtained when the micropayment data to be hashed is kept below 56 bytes. This is the case for hash chains, hash trees, and one-time signatures.

We now explain the findings in more detail. For the hash functions we see that there is a drop in the number of operations when the input data reaches 56, 120, 184 and 248 bytes. For both algorithms, the input message is padded so that its length is 64 bits short of a multiple of 512 bits (64 bytes). Padding always takes place and is performed by adding a single 1 bit to the message followed by as many 0 bits as are necessary. The final 64 bits are used to hold the length of the original input message. After padding and appending the length the data is a multiple of 512 bits and is processed in 512 bit blocks by the main algorithm. For example, a 440 bit (55 byte) input is padded to 448 bits, using one 1 bit and 7 zero bits, the length is appended, and then the

single 512 bit block is processed. However a 448-bit (56-byte) input is padded to 960 bits, because the single 1 bit must always be added. One bit alone brings the data to 449 bits, which does not leave 64 bits for the length within the single 512-bit block, so it is padded out to form two 512-bit blocks. The drop in the graph is due to an additional block having to be hashed at 56 bytes and every 64-byte increment thereafter. Other hash functions such as HAVAL [ZPS92] have an even bigger block size of 1024 bits. The lesson is to keep the input message size below the point where an extra block must be hashed due to padding.

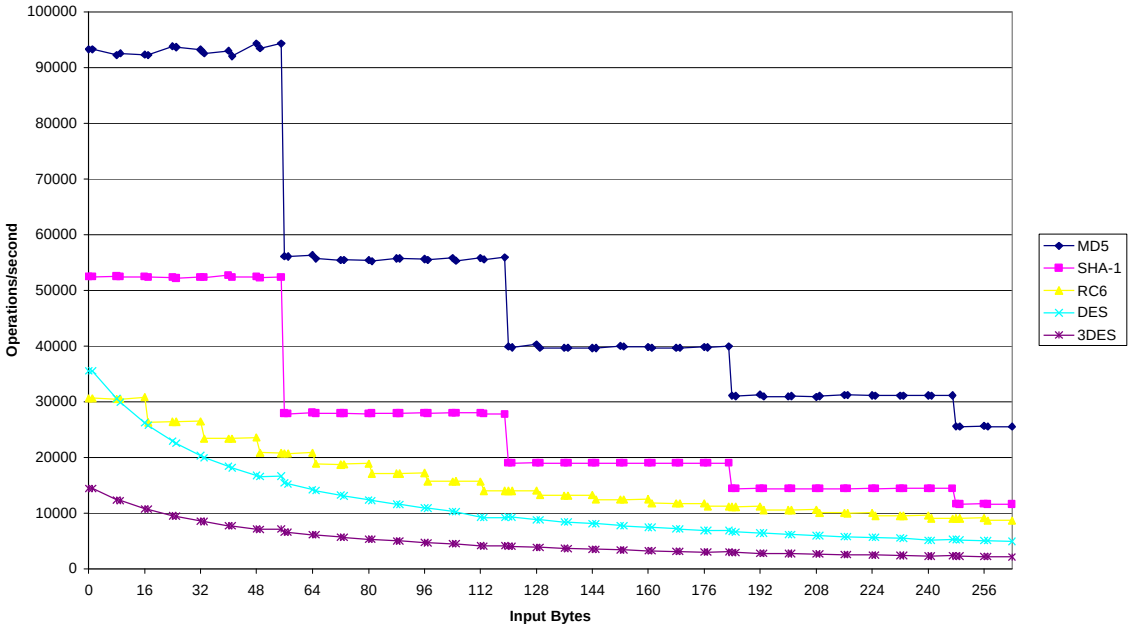


Figure 4-4 Hashing vs. Ciphering for Small Message Sizes

Another observation is that only 93,000 MD5 hashes are possible per second for a small message, whereas the average speed per 64-byte block in a long message was 115,000 hashes in Figure 4-1. Although there is no key setup with the hashes, time must still be spent initialising chaining variables, and the necessary padding calculated and performed. With a long message, this is performed in total once for all the blocks and introduces a negligible delay, whereas with just a single block the delay introduced has the identified effect on the overall speed. The performance of MD5 drops approximately 40%, 29%, 23% and 18% for each extra block respectively, showing the declining effect of the setup overhead as the number of blocks increases.

RC6 shows a performance drop every 128 bits (16 bytes). This is simply due to the input block size being 128 bits. Once the input message is 1 bit or more greater than a multiple of 128 bits, an extra block must be encrypted. We notice that after 160 bytes of input, increasing the input size has little effect on the overall speed. Our earlier hypothesis, based on Figure 4-3, that RC6 could be as efficient as a hash function is not true for small messages. This is due to the key setup delay. Therefore, RC6 should not be used as a replacement to a traditional hash function for micropayments.

DES turns out to be more efficient than RC6 for messages of 8 bytes or less, although it is unlikely that such small messages would be used in a practical scenario. An interesting observation is that DES and 3DES form smooth graph curves, while we would expect 64 bit (8 byte) step drops due to the 64-bit block size. The reason for this is that PKCS#5 [RSA99a] is used by the J/Crypto implementation. 1 to 8 bytes are always appended to bring the message length up to a multiple of 8 bytes. This allows the padding to be differentiated from the message in the final block when decrypted. Thus, an 8-byte message will be padded to become a 16-

byte message, as will a 9-byte message. A performance drop will occur between every *(8n – 1)* and *8n* bytes, where *n* is an integer. This is confirmed by our measurement data at 55,56 and 119,120 bytes.

4.2.4 Other Hardware Platforms and Implementations

Having obtained accurate cryptographic measurements for a Java implementation on a typical workstation, we compare these results with other implementations on the same workstation and with other devices. It is important to be able to estimate how many cryptographic operations a typical mobile device, PDA or smart card can perform, before designing a payment scheme that is expected to operate on these devices. We used a JavaCard as our smart card example. The JavaCard uses a JVM operating system to control access to all of the card's resources. The JavaCard JVM [Sun99] implements a subset of the original Java [GJS96]. Multos [MAO00] is an alternative multi-application operating system for smart cards, developed by Mondex International. The advantage of JavaCard is that multiple applications can be developed in a high-level language and be securely loaded onto the card at any time.

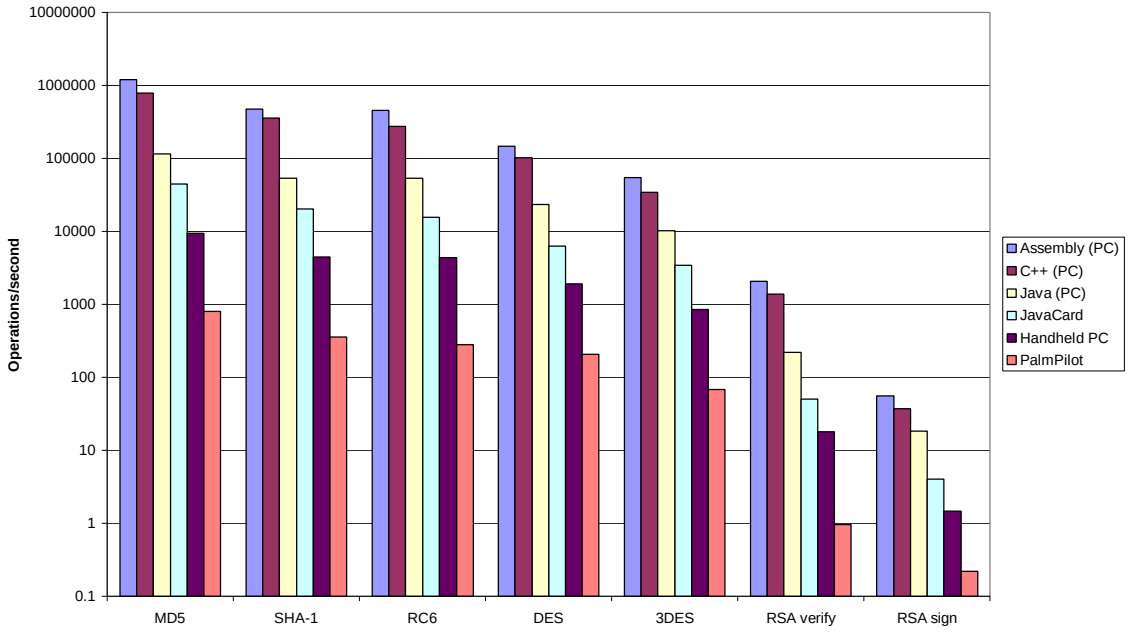


Figure 4-5 Cryptographic Algorithm Speeds for Various Implementations and Devices

Figure 4-5 shows a comparison of the number of cryptographic operations performed per second across different implementations and devices. Not all algorithms were available on all devices and, in some cases, results are estimated, as described in Appendix A. The idea here is to gain an appreciation for what mobile devices are capable of cryptographically, rather than an absolute benchmark.

The bar chart shows that native Assembler outperforms C++, which in turn outperforms Java implementations. The fastest algorithm on the workstation is the Assembler version of MD5 which can perform over 1.18 million hashes a second. Almost half a million SHA hashes are possible, with a similar number of RC6 encryptions. However, as uncovered in the previous section, RC6 will not be as fast as SHA for small messages. 147,000 DES and 53,000 3DES operations can be performed but only 2,000 RSA signature verifications and 55 RSA signature generations can be done. Clearly, public key cryptography performs very poorly compared to the other algorithms. The Assembler code is roughly 1.5 times the C++ speed. In turn, the C++ code is significantly faster than Java, ranging from 2 to 7 times faster depending on the algorithm. The Assembler MD5 hash speed is 10 times faster than Java, showing that despite JIT

compiler advances, native code speeds cannot be approached. Where speed is critical, an optimised Assembler implementation still significantly outperforms all other languages.

JavaCard speeds are surprisingly fast due to the use of a high performance version with cryptographic hardware accelerators, as detailed in Appendix A. This yields speeds that are only three times slower than on the workstation. Again, RSA suffers a further performance degradation, and is four times slower. However, JavaCards are used by a customer, rather than at a merchant or central server, and these speeds are very acceptable for client side operations. These speeds show that the perceived slowness of smart cards for cryptography is no longer an issue. Cryptographic accelerators may also be used on the workstation or heavily loaded servers to improve performance. For example, the fastest DES implementation is the DES Application Specific Integrated Circuit (ASIC) [WPRW+99] from Sandia National Laboratories. It has a throughput of 6.7Gps or approximately 105 million DES operations per second.

Commodity handheld computers and PDAs showed greatly reduced speeds. These portable devices have roughly the same compute ability as a mobile phone. All algorithms were two orders of magnitude slower for the handheld PC and three orders of magnitude slower for the PalmPilot, than on the workstation. RSA operations were very slow, requiring 5 seconds to generate a signature and 1 second to verify it on the Palm computer. With a heavyweight macropayment scheme, which processes multiple signatures at the client side in a single transaction, this may introduce an unacceptable delay. Even on the slowest device, hundreds of hashes are possible per second.

4.2.5 Public Key Alternatives

Having observed the computational slowness of RSA compared to symmetric ciphers we examine if a faster, more suitable alternative exists. Public key algorithms rely on the difficulty of a specific mathematical problem, relative to the input size, for their security. RSA is based on the integer factorisation problem, as is the Rabin scheme [Rab78, Wil80]. Other difficult tasks are the discrete logarithm problem, used in the Digital Signature Algorithm (DSA) [NIST94], and the elliptic curve discrete logarithm problem used in elliptic curve cryptosystems. Elliptic curves are defined over integers modulo a prime number, GF(p), or over binary polynomials, GF(2^m) [JM97]. Elliptic curves using GF(2^m) are slower than those using GF(p).

Schemes based on the same problem can be directly compared but it is more difficult to decide the key sizes that give comparable security for systems based on different problems. An estimate should be based on the time required by the fastest attacks known against each algorithm. The General Number Field Sieve (GNFS) is both the fastest factorisation method and the most efficient way to find discrete logarithms [Wie98]. GNFS requires approximately the same time to factor a 1024 bit RSA modulus as to perform a discrete logarithm with a 1024 bit DSA modulus. We therefore assume that 1024 bit RSA offers equivalent security to 1024 bit DSA.

Parallel collision search is the quickest method to find elliptic curve logarithms [Wie98]. Its low memory requirements allow it to be implemented much more cheaply in hardware than GNFS. It has been estimated that a 160 to 170 bit ECC key provides equal security to a 1024 bit RSA key [Cer97, Wie98, RY97], based on the time taken to execute GNFS in comparison to parallel collision search. The higher 170 bit estimate assumes use of a cheap hardware implementation of parallel collision search, which would allow ECC systems to be broken more quickly, and hence the larger key.

We used GF(p) with a key size of 168 bits, the only multiple of 8 between 160 and 170 bits, and compared it with 1024 bit RSA and 1024 bit DSA using Crypto++, as shown in Appendix A. DSA was found to be four times faster and ECC was three times faster than RSA for signature generation. This is partly due to the

precomputation that can be performed in both DSA and ECC but not RSA. However, RSA was over an order of magnitude faster than both DSA and ECC for signature verification, due to the use of small public RSA exponents. The same relationship is true for encryption. Therefore, unless there is far more signing than verification, which is unlikely in a payment scheme, RSA is more computationally efficient than either DSA or ECC. In addition, both ECC and DSA require the storage of extra system parameters which are approximately 500 bits and 2000 bits respectively.

The advantage of ECC is its short key size compared with RSA or DSA. The signature size of a hashed message is also small, requiring only 320 bits, compared to 1024 bits for RSA. For minimising transmission on a wireless link, or use of storage in a smart card, ECC is more suitable. DSA also has a small signature size of 320 bits, but the public key is the same as RSA at 1024 bits. In summary, overall RSA yields the best public key computational speeds and ECC offers reduced key and signature size. However, the computational speeds are still orders of magnitude slower than symmetric and hashing algorithms. It must also be noted that ECC is a more recent discovery [Mil85] than RSA and hence its security has not yet been analysed as closely [Kal98].

An even more recent proposal which has received some attention is the Efficient Compact Subgroup Trace Representation (XTR) [LV00] public key system. Its security is based on a variant of the traditional Diffie-Hellman [DH76] discrete logarithm system. Signing is up to 4 times faster than RSA, and the equivalent key size is 680 bits. However, initial measurements taken by the XTR designers show it to be 5.5 times slower for verification than RSA, and therefore not as suitable for payment systems where verification is the dominant online operation.

Using experimental results, we have shown how more efficient hashes and symmetric ciphers are over traditional public key cryptography. Acceptable cryptographic speeds can now be obtained using Java, but results vary depending on the JVM and implementation optimisations used. Traditional hash functions are more efficient than ciphers for messages under 250 bytes. Even on small portable devices, thousands of hash functions can be performed per second, suggesting that hash functions form an essential efficiency component of micropayments. Where public key algorithms must be used, RSA is the most suitable choice. Based on these observations, we can now directly compare the computational cost of different micropayment schemes and design our own efficient micropayment protocol.

4.3 Micropayment Performance Evaluation

The majority of micropayment schemes examined have never been implemented and, therefore, performance results do not currently exist. These schemes have not been critically compared against each other as usually only the improvement a scheme makes over an earlier related technique is highlighted. In this section, we critically evaluate the important schemes from each of our classification categories against each other. We compare the storage, communication, and computational costs of each.

The amount of storage required by micropayment key and payment material is important when designing for small mobile devices. Equally important to the scalability of the scheme is the amount of data that the vendor must hold and process for every customer. Our communications measurements are concerned with the size of the payment messages sent over a wireless link, the number of messages required, and the online involvement of a distant third party. The payment traffic adds an additional signalling overhead which should be kept as a small percentage of the total user traffic, especially if bandwidth is limited. Finally, and perhaps most importantly, the amount of computation required to make and verify each payment will limit the total number of transactions that can be processed per second.

A detailed analysis and comparison of each scheme allows us to see which techniques work well in each of these three performance areas. From this we can then accurately and quickly select the useful components for our own multi-party schemes. In addition, the study allows us to validate each of our three micropayment contributions put forward in Section 3.3.

Since implementations do not exist, we use the published protocols to predict the cryptographic functions and message sizes that would be used in their implementation. That is, we extend each scheme from a high level description to actual methods and messages, using a uniform format across all designs. The implementation assumptions made and the resulting data are summarised in Appendix B.

4.3.1 Storage

Every payment system will require a number of permanent storage bytes for the security keys and actual payment instrument material. For the user, this is important where mobile PDAs or smart cards with limited storage capacity are employed. Every vendor and broker will have a storage cost per user, and the size of this will affect scalability and performance issues. For example, anonymous electronic cash requires the broker to record the serial number of every coin spent in the system, until it expires. In an offline micropayment, the vendor will have to store the payment until it is redeemed.

We divide the storage requirements into two parts. The first part is the space occupied by certificates, public and secret keys. These are used for authentication and to secure communications. If a scheme requires all parties to have a public key certificate, then the space required must be allowed for. Secondly, we consider the space occupied by the payment instrument. Temporary buffer space is not included. If the payment instrument can be generated dynamically, such as an electronic cheque, then there is no storage space required for it by the user. In contrast, every electronic coin must be stored. The storage costs were calculated for a single user who buys from a specific vendor, with a broker providing services to both.

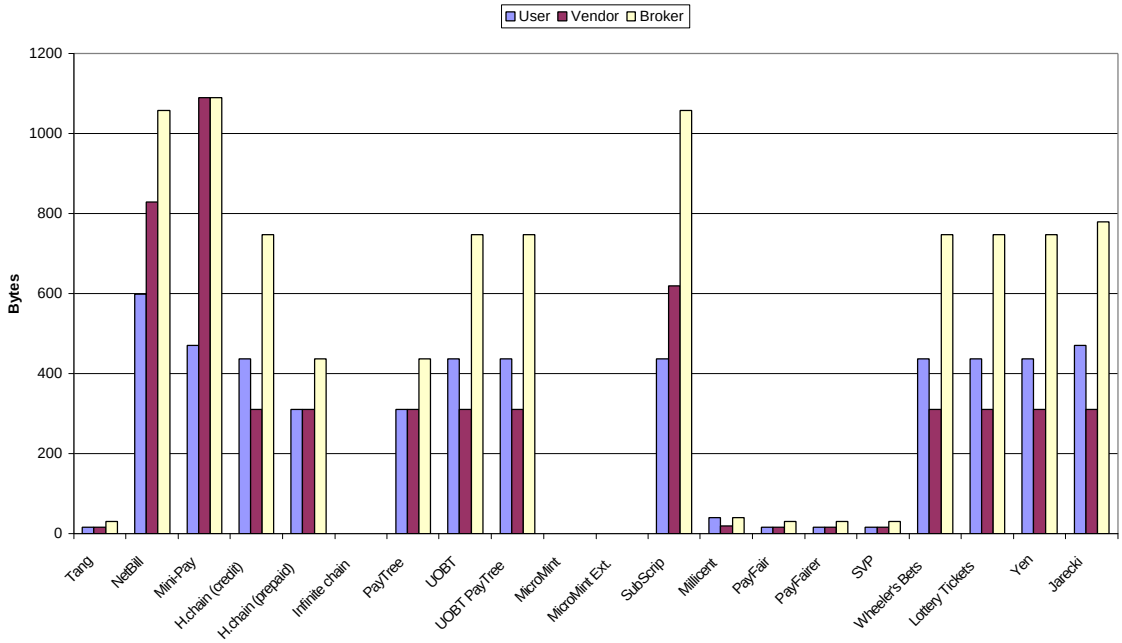


Figure 4-6 Certificate and Key Storage Space required by each Entity

Figure 4-6 shows the size in bytes of the key material required at each party. No scheme requires more than 1,100 bytes of key material at any party. However, even this is considered too large by some manufacturers, as demonstrated by their design of small mini-certificates [WAP00] for use in mobile PKIs. The size of X.509 certificates, as used in SET, has also been highlighted as a problem. In addition, current smart cards,

such as the JavaCard used in our prototype, only provide 8 to 16 Kbytes of permanent storage which must store the application code as well as all keys and payment material. In such environments, it is always beneficial to reduce storage.

It can be seen that schemes using public key certificates require far more storage than secret key schemes. A 1024 bit RSA public key certificate is approximately 310 bytes while a symmetric key need only be 16 bytes. However, digital signatures verifiable with certified public keys provide non-repudiation, allowing an audit trail to be kept. This is not possible with uncertified shared secret keys. Shared secrets also result in central deposits of all the secrets in the system, a potential security weakness.

The bar chart shows that the secret key schemes all require approximately the same storage; 20 bytes or less are needed at both the user and vendor, and 40 bytes or less at the broker. However, Millicent shares a secret between the user and vendor. Thus for every different vendor the user visits, he will need an additional secret, which is not illustrated in the chart. This is unlike certificates and shared broker secrets, the number of which remains constant. A Millicent user who has scrip for 16 vendors or more, actually requires more key storage than certificate storage for hash chains. The hash collision based MicroMint and its derivations require no key material at all.

We note that the online schemes using certificates, and the public-key based schemes require more certificates than the hash-based schemes. With SubScrip it is assumed that certificates are needed for the vendor macropayment, yielding a similar storage overhead. The probability based schemes use hash chains if payment is actually necessary and therefore have the same storage overhead as them.

When comparing credit-based hash chains, which need user certificates, to prepaid ones, requiring only broker certificates, the vendor storage requirements in both are identical. In both cases, the vendor only needs to store the broker certificate, as once a user signature is verified on the commitment, the user certificate is no longer required. In a credit scheme, the user will have to store the private key corresponding to a user certificate. The broker will also have to store and administer user certificates in a full public key infrastructure (PKI) [Lan98]. This is a huge task with thousands or millions of users. We discuss the drawbacks of user certificates in Chapter 5.

4.3.1.1 Payment Material Storage

When considering the storage space for payment material, we want to show the requirements before and after a number of micropayments have been made, rather than for a single payment. Our calculations are based on a user holding €10 worth of payment material which can be used to make 1,000 payments to a specific vendor, as might occur during a telephone call. We also include any additional storage required after the first payment, such as the current index in a hash chain. For the vendor, we show the maximum storage required during or after receiving 1,000 payments from the user. Finally, for the broker, we show the storage costs after 1,000 payments have been redeemed by the vendor. The broker must maintain a database to prevent double redeeming and, perhaps, a transaction log. For a credit based scheme, proof of a transaction must be kept as a user could otherwise deny it. With a prepaid scheme, an audit trail is not necessary, unless the value of unspent prepaid tokens can be re-claimed by the user. Figure 4-7 shows the storage space required by each entity for payment material.

The huge storage costs of the hash collision schemes is immediately obvious, with 80,000 bytes required to store the coins. Macropayment electronic cash schemes, with bank-signed coins, have an even greater storage cost. The saving in having no key material has been far offset by the coin size. However, Figure 4-7 only

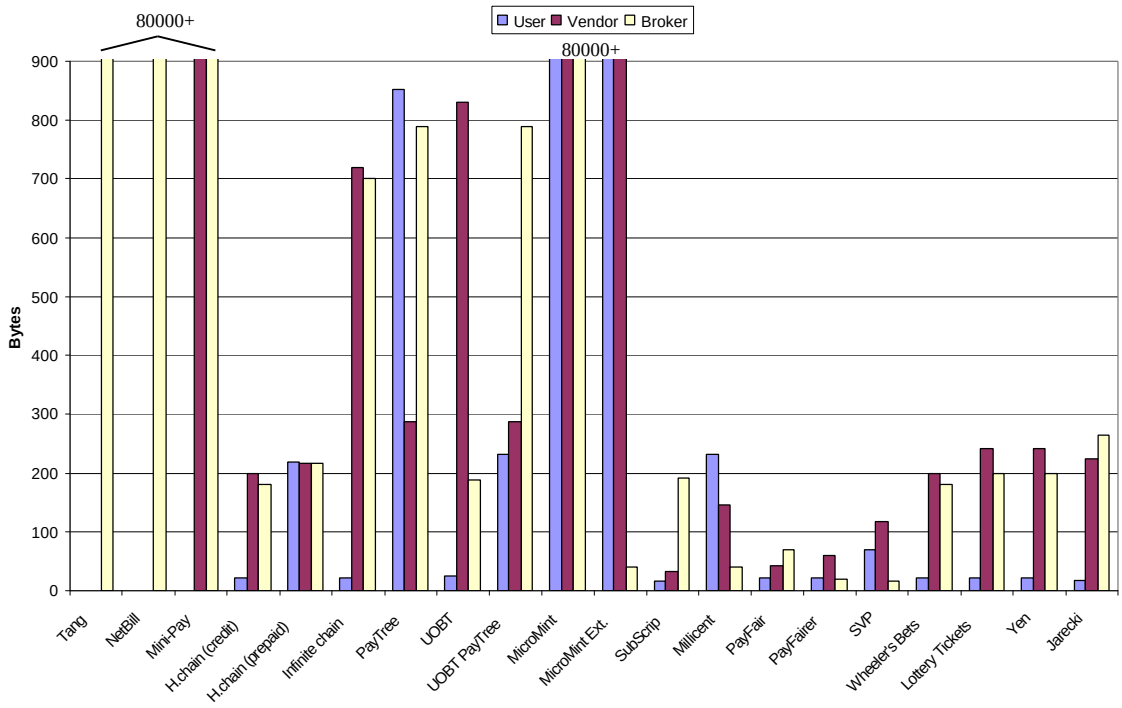


Figure 4-7 Storage Space required by Payment Instrument Material

shows a scenario with a single vendor. If 1,000 uncorrelated micropayments were made to 1,000 different vendors, the MicroMint user storage space would be only four times greater than that for a credit hash chain. In turn, the storage space required at each vendor would be less than half that required by the credit hash chain. However, in most applications repeated payments to the vendors is more common.

The online schemes only require storage at the broker, since the payment is dynamically generated by the user and cleared in real-time. The transaction log results in a large storage cost for both online and public key schemes, and results in similar storage costs to full macropayment systems.

Credit hash chains require very little user storage as the chain and commitment can be dynamically generated as required. In contrast, with prepaid, the broker signed commitment, of approximately 200 bytes, must be kept. The increased flexibility of the hash chain derivatives increases their storage costs. The extra storage for the infinite chain is solely for the one-time signature, showing it to require three times more storage for an extended chain than a new one. The advantage of the infinite chain is a computational one.

PayTree increases the user storage to over 800 bytes due to the extra hash chain costs. However, this is still more compact than using separately signed hash chains. In contrast, UOBT places this cost on the vendor who must store all the leaves of the tree, or anchor hashes, after receiving them in the first payment. Our combined UOBT-PayTree significantly reduces the extra user and vendor storage imposed separately by both.

The probability payments use a credit hash and have the same overhead. Jarecki requires slightly more storage at the broker due to the need to keep track of the amount spent from the polls received. SubScrip has minimal payment storage, but has greater key storage requirements. Again, the secret key based schemes have the minimum storage. The exception is Millicent, where the user payment material is about 250 bytes, which is greater than both credit and prepaid hash chains. They require less than 100 bytes in total for both keys and payment. Otherwise, secret key solutions require less than 100 bytes in total for both keys and

payment. The next best are hash chains, which require between 300 and 800 bytes in total at each party, most of which is for certificates. The total storage costs vary widely between schemes and will impact not only on limited capacity devices but can also mean the difference between holding a payment in memory for fast processing, or retrieving it from slower disk storage.

4.3.2 Communications

When considering the communications cost of a payment scheme, the size, length, and number of messages sent between parties must be calculated. When paying for the volume of traffic transported, or if making frequent payments, the signalling overhead due to the payment process should be kept small relative to the payload sent. If payment is made to or from a mobile device over an air interface, with limited or scarce bandwidth, the volume of payment messages should also be minimised.

We differentiate between the first payment and subsequent average payments. The first micropayment often has an extra communications and computational cost in initialising payment material so that subsequent payments are less costly. Typically, the first payment might use a digital signature and certificate. If several correlated payments are then made which do not have this initial overhead, a saving can be made. Other schemes, such as MicroMint or Millicent, have the same communications cost for both the first payment and further payments.

Our calculations assume that the first and subsequent payments are for one cent, the basic monetary unit. The total message traffic, in bytes, between each entity is shown. Where there are several inter-party messages, such as a 3-way handshake, the total traffic is shown, but the number of messages can be ascertained from Figure 4-10. In the bar charts, a bar which is split into two different shaded parts indicates an online communication with a third party, which imposes an additional delay.

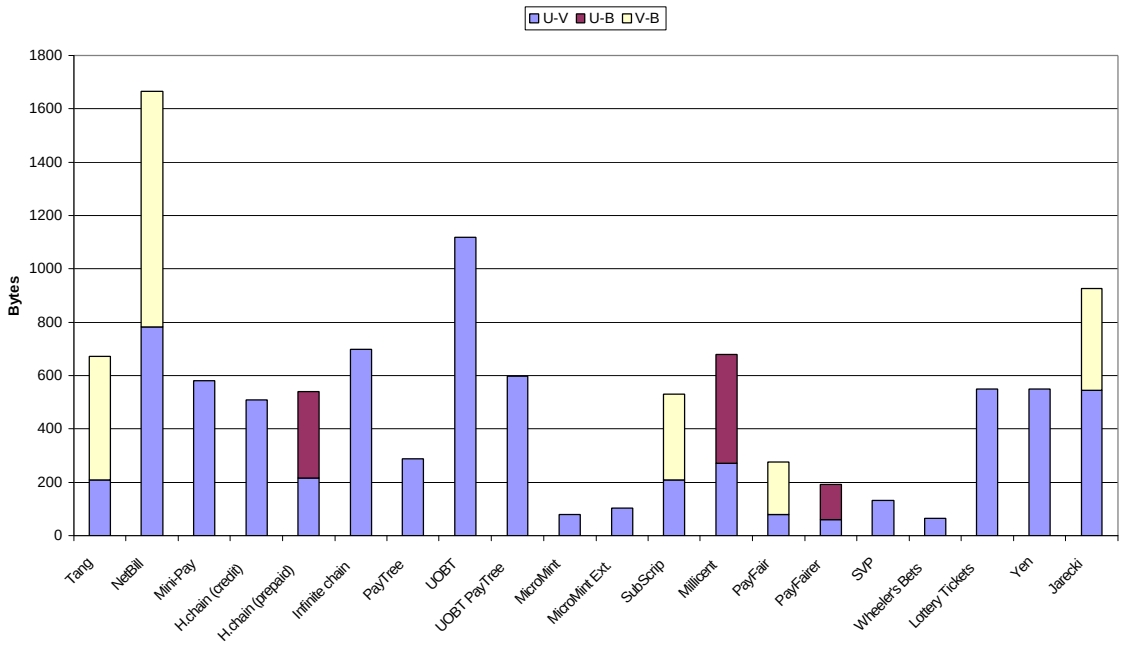


Figure 4-8 Communications Bandwidth used during First Payment

Figure 4-8 shows the communications bandwidth used during the first payment to a new vendor. The majority of schemes require less than 600 bytes. Hash collision schemes are the most efficient provided only one coin is sent. If seven MicroMint coins are sent, in order to pay seven cents, then this requires 560 bytes,

which is the same size as an initial hash chain payment. Wheeler's coin flips also require minimum communications, as a real payment is only rarely made.

The secret key schemes show mixed results. They are all online except the smart card based SVP. PayFair and SVP require less than 250 bytes but SubScrip and Millicent require more than double that, due to the macropayment in SubScrip and the scrip length in Millicent. Hash chains and probabilistic schemes are more efficient than these, and both use a signature on a new chain. If this is a user signature then the matching certificate, of 310 bytes, must also be sent. This explains how both credit and prepaid chains transmit a similar amount of data even though we assume that the prepaid chain is bought online from a broker. An online connection is not necessary if the vendor is known in advance.

PayTree gives a surprisingly good result, as the tree's chains are vendor specific and are bought offline. Our UOBT-PayTree almost halves the bandwidth required by UOBT, due to the removal of the need to transmit all the final hashes of the chains to verify the signature.

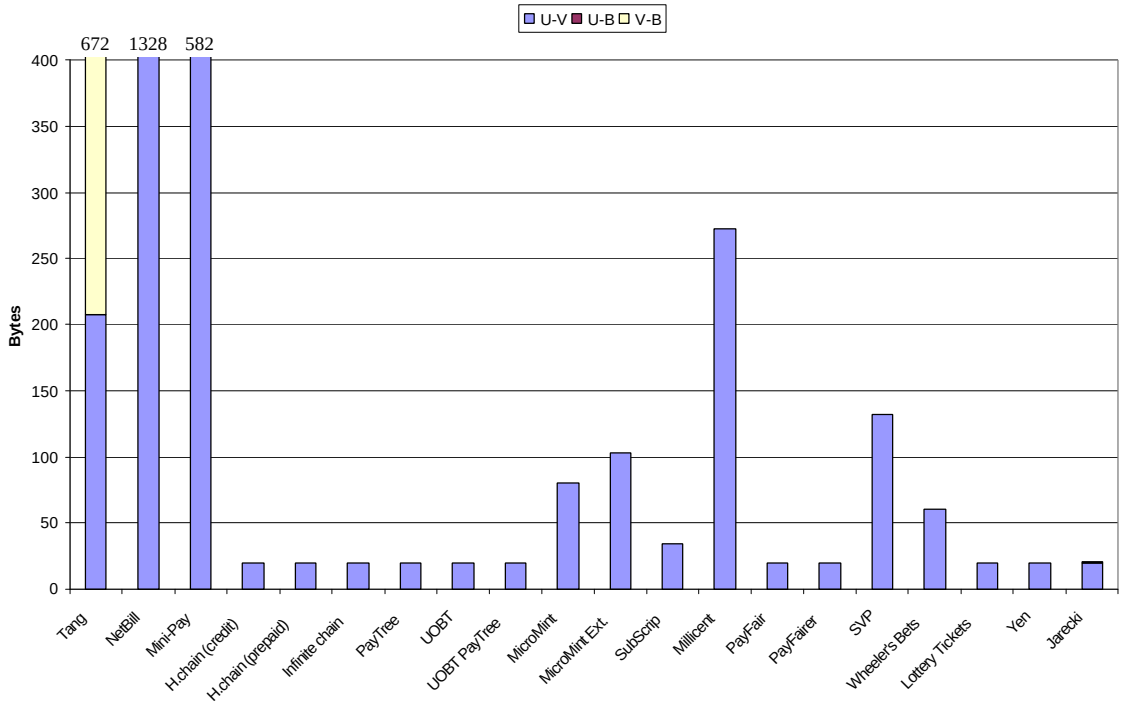


Figure 4-9 Communications Bandwidth used during Average Payment

Figure 4-9 shows the size in bytes of payment messages sent during an average payment, subsequent to an initial one. An important observation is that all the schemes which use hash chains, including the probability schemes and PayFair, only require a single 20-byte hash value to be sent per payment. This is likely to be the minimum message size possible for secure micropayments. Including the first payment, this brings the average message size for 10, 100, and 1,000 credit hash chain micropayments to be 69, 24.9, and 20.40 bytes respectively.

SubScrip also uses very small messages of only 34 bytes, but is not secure from eavesdroppers. The collision schemes require exactly the same as the first payment, less than 100 bytes. Millicent has a surprisingly high cost of over 250 bytes, more than 12 times that of hash chains. This is due to the large number of fields in scrip and having to issue scrip change. Finally, Tang, NetBill and Mini-Pay require over 400 bytes, which may be suitable for once off small payments but not repeated micropayments.

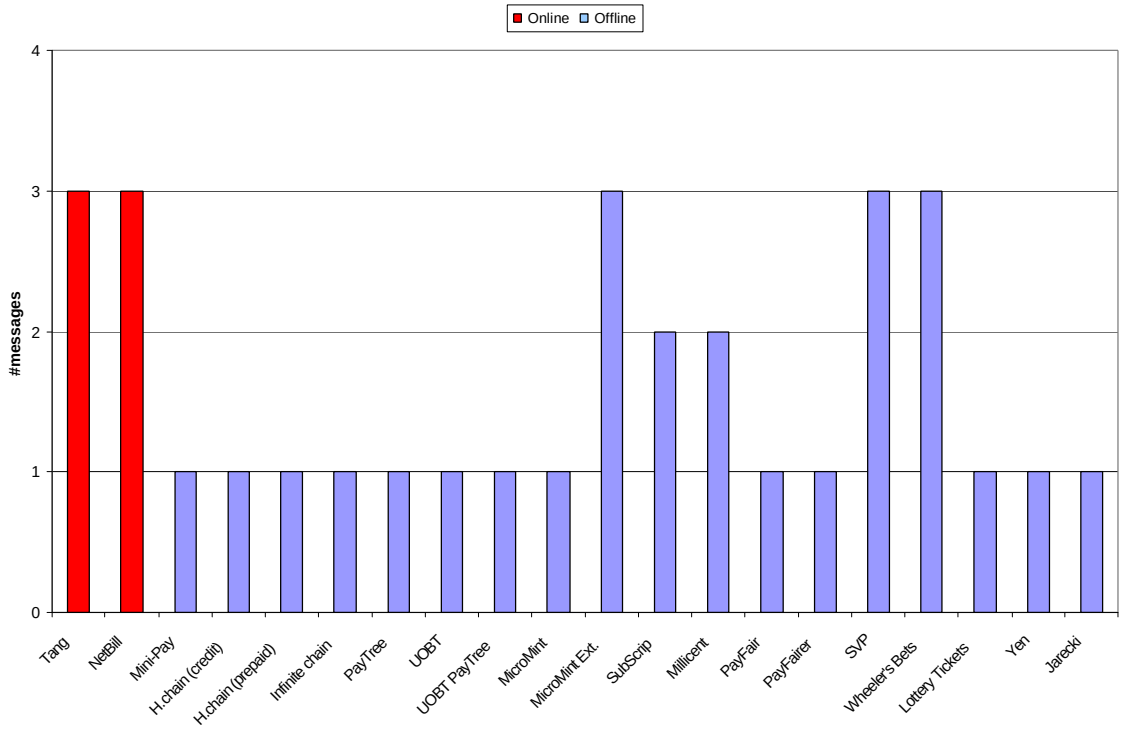


Figure 4-10 Number of Messages sent during an Average Payment

Figure 4-10 shows the number of messages sent during each payment following the first. Only Tang and NetBill are online for every payment. All schemes using hash chains require only a single payment message. This is especially suitable where transport services are being paid for, as payment can be sent along with the payload, with no need for a vendor reply. In contrast, both Millicent and SubScrip require token change for following payments. Therefore, secret key schemes alone are not as suited for paying for transport services. A coin flip requires a three-way handshake, as does SVP and MicroMint extensions. Polling messages in Jarecki's scheme are so infrequent that the average number of messages is only 1.0025.

In terms of minimising communications, hash chains are the most efficient after an initial first payment. On the other hand, whilst secret key schemes combined with hash chains allow small messages for the first payment, they are online, unlike a credit hash scheme.

4.3.3 Computation

The amount of computation that must be performed by each party during payment is the third efficiency component of a micropayment scheme. Computational requirements limit the maximum number of transactions per second at all parties, and may be a bottleneck. Vendors will typically need to handle up to many thousands of transactions concurrently. Similarly, a user device needs to be able to perform all the payment processing, perhaps more than a payment per second, in addition to normal applications. To allow both scenarios computation needs to be minimised.

We use our measurements from Section 4.2 to calculate a computational relationship between the different cryptographic operations. The method and data used are given in Appendix B. Since devices of differing computational capabilities will be used by each entity, an exact payment computation time cannot be computed. For example, the computational differences between a PC and a PDA are highlighted by Figure 4-5. Therefore, the computational cost of each cryptographic operation is expressed in terms of the number of hash operations, the fastest computation, that can be performed in the equivalent time. An SHA hash is set to take one unit of time with everything else expressed relative to this. If desired, the timings for each payment

scheme can be quickly calculated given the performance time for SHA on a specific platform, as in Section 4.2.

Figure 4-11 shows the computational cost of the first micropayment to a new vendor for each online party involved. As expected from the results in Section 4.2, those schemes with signatures have the highest load. The collision schemes and secret key schemes have virtually no overhead compared to those using signatures.

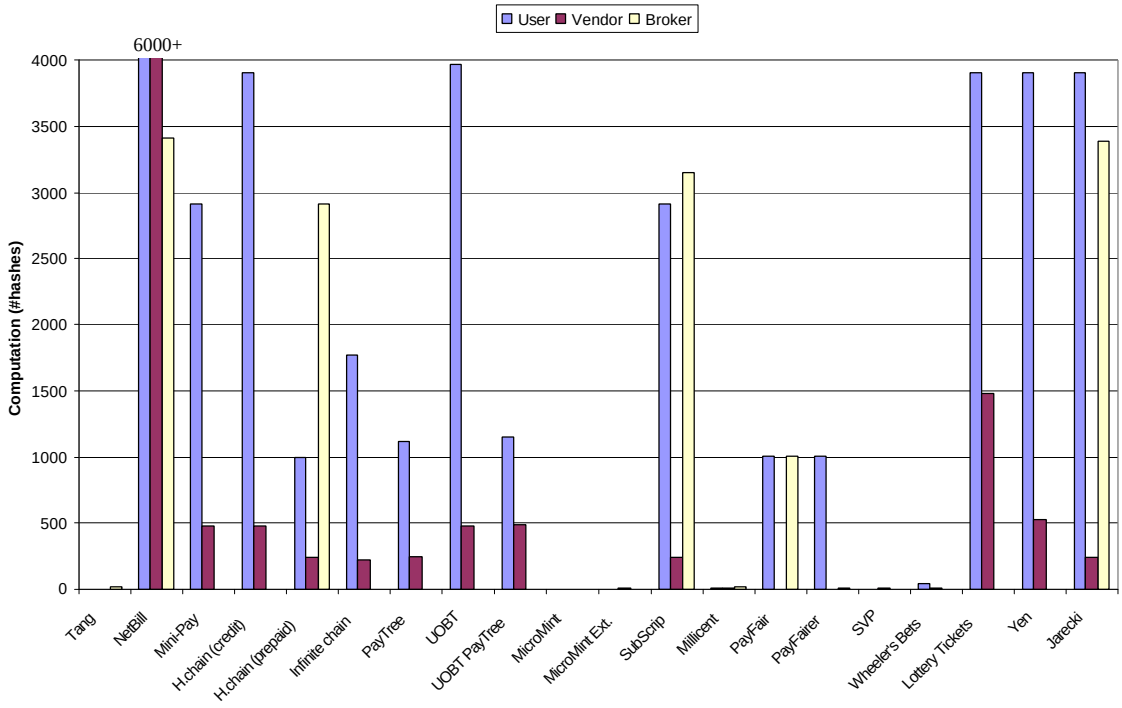


Figure 4-11 Computation Performed per Party for First Payment

Hash chain schemes also have the overhead of generating a new chain, which is included in the chart. In practice this computation can be done offline, and each user could have a collection of new chains ready to be signed. The computation shown for PayFair and our PayFairer, is for chain generation. Credit hash chains have the user signature overhead, while, with prepaid, the signature is performed online by the broker.

Of the hash chain derivatives, PayTree performs the best, followed closely by our UOBT-PayTree. The single signature on multiple chains ensures that a new one is not required for each new vendor. RSA signature verification is seen to require less than a quarter of the computation needed to generate a new chain of length 1000. Mini-Pay performs better than hash chains for the first payment, because it has no chain generation overhead. Hash based lottery tickets and Yen's improvements are more expensive for the vendor, due to the vendor chains. Yen's improvement requires approximately one third of the vendor computation for the first payment. With SubScrip, there is an initial macropayment overhead, that is less than the first payment in Jarecki and every payment in NetBill.

Figure 4-12 shows the computational costs for an average payment. The majority of schemes show a three orders of magnitude performance improvement over Figure 4-11. This is a key feature of micropayments. The performance of each class of scheme is now briefly compared.

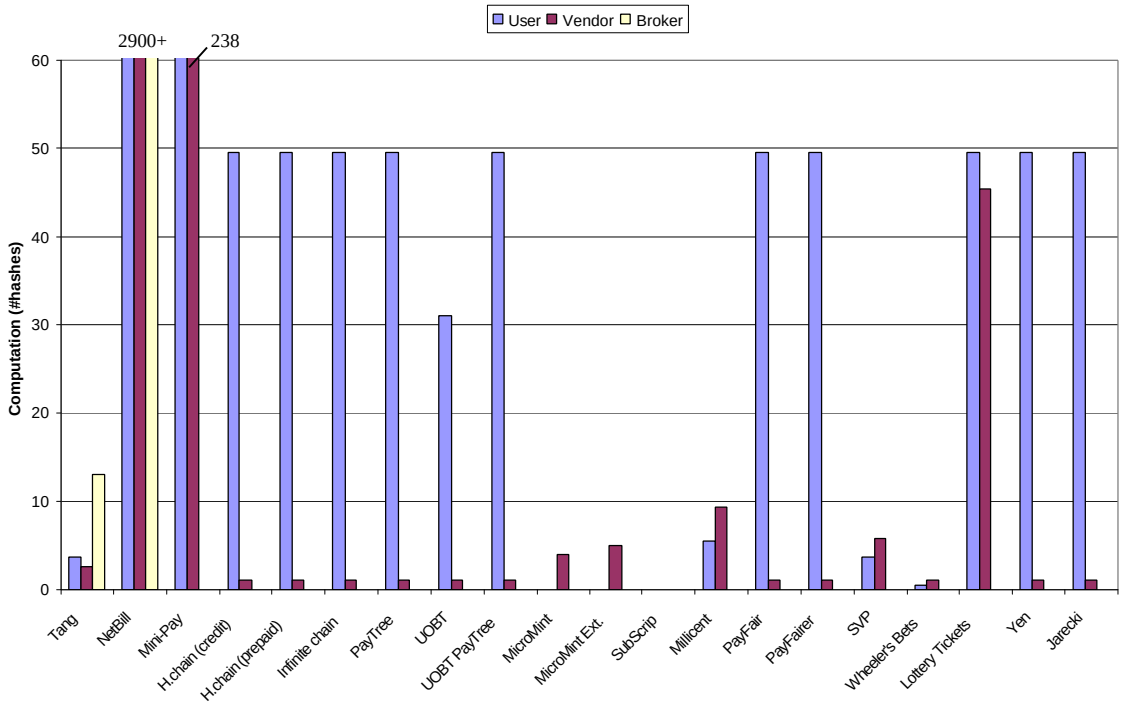


Figure 4-12 Computation Performed per Party for Average Payment

SubScrip has no cryptographic overhead but its use of unprotected account identifiers also offers no security against eavesdroppers. Possible overheads due to database access are discussed in Appendix B. The hash chain schemes minimise the vendor computation to a single hash calculation. However, the user computation will be more unless the entire hash chain is cached in memory, which would require 20,000 bytes, more than that available in current smart cards and small PDAs. We assume that key parts of the chain, every 100 hashes, are cached which keeps the average user hashing per payment to below 50. Both credit and prepaid chains have the same performance for the average payment. UOBT reduces the number of user hashes without any caching, and its performance could be further improved if caching were introduced.

In contrast, the hash collision schemes require no user computation at all but increase the number of hashes performed by the vendor. Wheeler's coin flips only require user hashing when an actual payment is made and since this occurs infrequently, the user cost is less than a single hash while the vendor cost is just over one. Millicent and SVP also show good performance results, with less user computation than hash chains but more vendor hashing due to the size of the scrip and request. Lottery tickets have the highest vendor costs of all the schemes, except those that use full signatures for every payment. The vendor cost is approximately the same as the normal user hash chain cost due to the need to rehash uncached parts of the chain.

The computation for a first payment varies enormously from a few hashes up to almost 4,000 hashes. Each following payment is reduced to 50 hashes or less, with hash chains minimising the payee workload where the most payment processing occurs.

4.4 Macropayment and Micropayment Performance Comparison

The performance of five existing macropayment systems is now compared with three representative micropayment schemes. It is shown how unsuitable the macropayment schemes are for repeated payments, and how their communications and computational requirements are up to four orders of magnitude greater than average micropayment schemes.

To represent macropayment protocols we chose SSL and WTLS as secure transport protocols, SET for credit/debit card payments, Mandate II e-cheque for offline electronic cheques, and fully anonymous electronic cash with blind signatures as deployed by Ecash Technologies, formerly DigiCash. SSL, and its wireless counterpart WTLS, are currently used on the majority of e-commerce sites for enabling secure transactions. However, they are not payment protocols, but rather provide a means for authenticating parties and securely transporting data between those parties. Typically this data is user credit card information. The performance estimates are based on the referenced published protocols. Any assumptions made and the resulting calculated estimates are detailed in Appendix B. Credit hash chains, MicroMint and Millicent were chosen to represent typical micropayment schemes based on hash chains, hash collisions, and shared secret keys respectively.

4.4.1.1 Communications Comparison

The size of communications traffic, between the user and vendor and between any party and the broker, for both the first payment and following ongoing payments, is shown in Figure 4-13. The chart immediately contrasts the much lower communications overhead of micropayments compared to the macropayment systems. SET sends the most traffic, with over 6,000 bytes transmitted for each payment. There is a flow of 3,000 bytes from the user to the vendor, which will add a noticeable delay over a slow wireless link. For example, it would take 2.6 seconds alone to transfer this user payment traffic over the 9600bps wireless link provided by current GSM phones, assuming that no other application was simultaneously sending data.

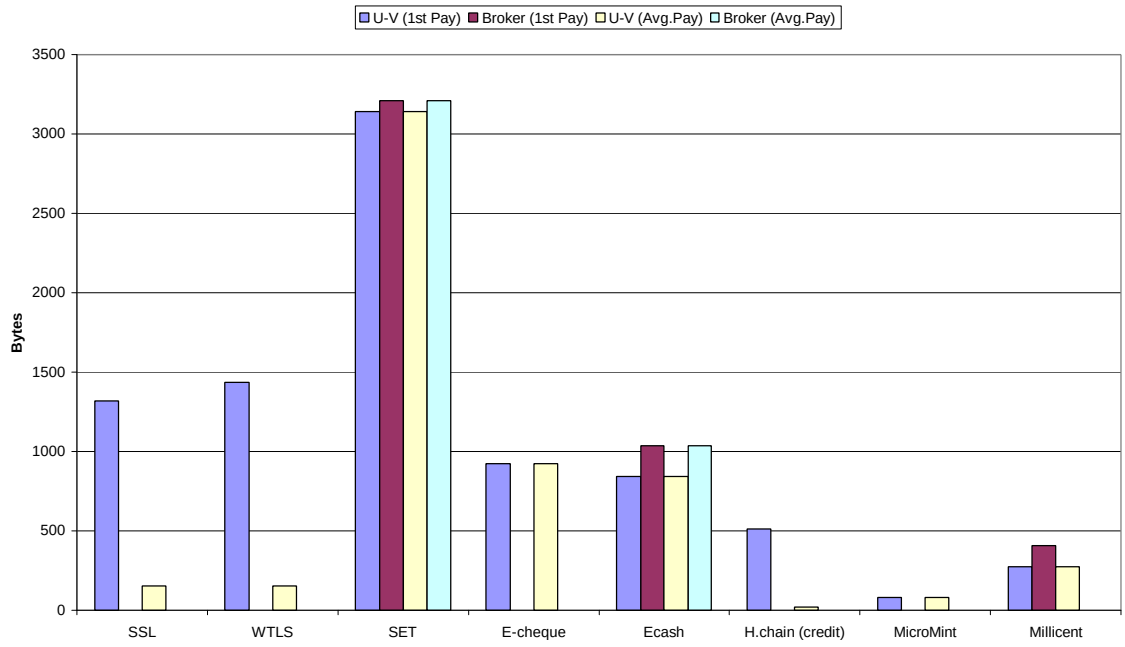


Figure 4-13 Macropayment and Micropayment Communications Overhead Comparison

Ecash is also online for each payment, and requires approximately 1,000 bytes to be sent between each party, even when only five coins are used, as we have assumed. Each coin is 128 bytes, and therefore the message sizes may get much bigger if a larger number of coins are required, a likely scenario since change cannot be issued if anonymity is to be preserved.

SET, SSL and WTLS all use 6 messages in the initial payment. With SSL and WTLS all 6 are sent between the user and vendor, incurring additional delays over a wireless link. Although WTLS was designed as a lightweight wireless version of SSL/TLS, we have assumed that mutual certificate-based authentication occurs, unlike SSL where only the server is usually authenticated. This assumption is based on the vision

that every WAP device will hold a user certificate, allowing mobile wallet functionality and authentication to personalised mobile services. An SSL/WTLS merchant will normally verify credit card details online through existing financial networks, and this will introduce an additional delay and communications cost not considered in the chart.

The initial overhead in the secure transport protocols is in establishing a new session where signed messages are exchanged. SSL and WTLS show a significantly lower communications overhead for ongoing payments. This is due to the shared secret keys, established during the initial handshake, which can be used to encrypt payment material efficiently. However, no real payment system is provided by these protocols; there is nothing to stop the merchant abusing a received payment card number or passing it onto others to abuse; there is no proof that the user agreed to the specified amount, or indeed that the user is actually present in the case of SSL with no user authentication.

SSL allows secure sessions to be resumed, based on knowledge of the shared secret keys originally established. The default SSL session timeout is 300 seconds with Windows 2000. However, many large Web sites, which use a pool of replicated Web servers, cannot re-use cached sessions, as a resume request may not be passed to the same Web server process as before, which has the keys cached [APPS00]. This can result in a new SSL handshake for every request to the site.

The e-cheque scheme can be offline due to the use of smart cards to prevent cheating. For every payment it has a similar communications cost to Ecash, and several hundred bytes less than a first payment with SSL/WTLS. However, requiring over 900 bytes to be sent per payment is too great for rapidly repeated payments from a mobile.

The micropayment schemes, each of which was discussed independently earlier, all send less than 510 bytes for a first payment, and less than 280 bytes for ongoing payments. Only 20 bytes are sent for a hash chain payment. For communications over a wireless link, the micropayment schemes are more suitable than the macropayment systems.

4.4.1.2 Computation Comparison

The efficiency differences between macropayments and micropayments become more pronounced when considering computation. Figure 4-14 shows the computational cost for an initial payment, and for any further ongoing payments after that.

SET has the poorest performance, requiring the equivalent of over 4000 hashes at every user for each payment. This becomes even more hefty at the vendor and broker requiring 13,000 and 16,000 hashes respectively. This broker load equates to 3.3 operations per second in Java on a commodity 400MHz PC. When compared to the 53,590 hash chain payments that can be processed in the same time on the same machine it seems almost incapable. Ecash is also computationally heavyweight at the vendor and broker, with every payment requiring more than 2900 hashes. However, only the equivalent of 310 hashes are required at the user for every payment, the overhead of encrypting the coins with the bank's public key. Although e-cheque is offline, it has a cost of over 3100 hashes for every cheque, due to the signatures involved.

SSL performs favourably and, at the user, this is due to not performing any user authentication that would require a digital signature. In the scenario outline above WTLS does perform user authentication and it is this signature that makes it over six times more computationally expensive at the user. A credit hash chain must

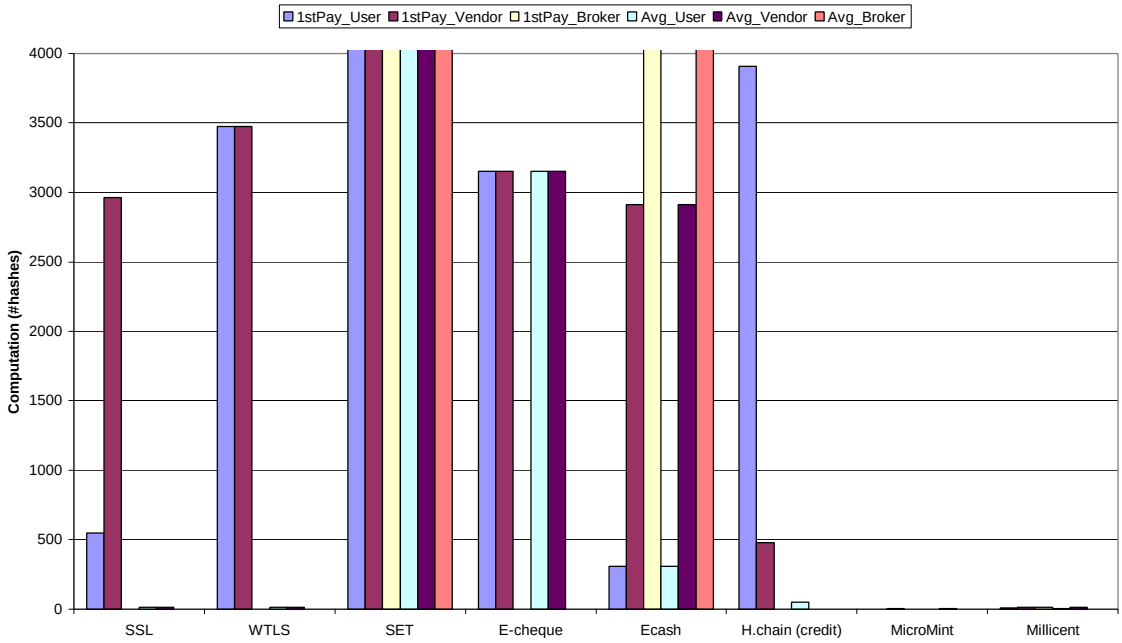


Figure 4-14 Computational Cost Comparison between Macropayment and Micropayment Systems

also employ a user digital signature, in addition to constructing a new hash chain, which gives it a greater cost than either SSL or WTLS for the first payment. However, a hash chain user need not interact with the vendor, unlike SSL and WTLS, and, therefore, if the vendor is known in advance these computations can be performed offline, eliminating their cost.

MicroMint and Millicent have the lowest overhead for a first payment, both requiring fewer than 10 hashes. While SSL and WTLS use efficient symmetric encryption of payment details after the initial setup, they cannot offer the security or guarantees of a real payment system. Nor is overhead incurred in verifying the payment instrument through the banking network considered in the chart. In contrast, we see that the micropayment schemes are offline, provide security of payment, and non-repudiation in the case of hash chains. These come at about the same computational cost as SSL, but offer the advantages of a real payment scheme. In comparison with the true macropayment systems, the micropayment schemes are three orders of magnitude more efficient on average.

4.5 Performance Analysis Conclusions

Real macropayment systems, such as SET, e-cheques and anonymous ecash, bestow a heavy communications and computational cost on all parties. The message sizes are too large to send regularly over slow wireless links. Without the use of a smart card, the schemes are online with a distant third party. While the computational requirements could be handled by installing additional processors at each entity, micropayments can be used for small payments at a fraction of the cost. Hash chains require a mere 0.00078% of the computation compared to SET to verify each payment at the merchant, and 0.003% of the computation compared to electronic cash or cheques. This will allow at least three orders of magnitude more payments to be handled at a merchant for the same price. This is important for the scenario of Chapter 1 where there are many small merchants providing network access over local wireless links. These entities will be able to set up services and receive payment using low-price commodity PCs, rather than requiring fast expensive servers. This will give anyone the power to be a network operator, and will pave the way for fast, cheap, local wireless access. Larger network operators and central core network operators will also benefit from using a micropayment scheme, over traditional macropayments, by being able to handle a larger number of call setups and simultaneous calls per second.

We have shown how macropayment techniques and the use of public key or online connections for each payment do not scale for frequent payments. The prediction that NetBill, Tang, and Mini-Pay do not scale for repeated micropayments is confirmed by analysis. Similarly, authentication schemes for distributed systems, such as Kerberos, are not suitable for use in a micropayment scheme outside a local Intranet environment. They depend on centrally administered shared secrets with online verification.

The micropayment performance charts show how different property tradeoffs result in varying performance results. The different micropayment schemes are basically moving the performance improvements and flexibility from one area to another. For example, the use of prepaid vendor independent tokens, as in MicroMint, greatly increases storage but reduces the first payment overhead compared to other schemes. In addition, by increasing storage, MicroMint minimises user computation, while credit hash chains have an initial user computation but minimal storage. The optimal scheme for a chosen environment depends on the properties of that environment. MicroMint would not be suitable for smart card applications, because of the storage requirements, but hash chains would be. The empirical analysis quickly highlights the performance characteristics of each scheme, and the survey in Chapter 3 examined other properties such as security and risk model.

In respect of storage, the secret key schemes are optimal. Hash chains have an acceptable storage need, especially for a user device with limited storage capacity. Hash collision schemes require more storage and are not suitable for such devices. In respect of communications overhead of an average payment, hash chains are the most efficient. It is important that only a single message is required, while secret key and probabilistic schemes require two or even three. Without a smart card, secret key schemes are online for the first payment. Although larger messages are used, schemes using credit hashes are all offline for the first payment. Prepaid chains and Millicent are online if the vendor is not known in advance, which is dependent on the scenario.

The computation required for an average payment is very acceptable for all micropayment schemes. Fewer than 50 user hashes are required, which are easily performed by all devices used in our measurements in Section 4.2.4. A user is unlikely ever to make more than one or two simultaneous payments. Only a single vendor hash is needed for hash chains and probabilistic schemes. While secret key schemes require more vendor hashes, they are kept to fewer than 10.

Overall hash chains provide the best communications and computational performance. They are best suited for a scenario with computationally lightweight user devices with small storage and limited bandwidth, and vendors who have to process a large number of payments per second. In addition to performance, the security properties are stronger than those of the next best performer, secret key schemes, where non-repudiation cannot be provided due to the shared nature of keys, and where a copy of all security keys must be kept centrally.

Implementation decision issues became evident when estimating the performance of each scheme in Section 4.3. For example, the tradeoff between re-computation of a user hash chain and caching of key points in that chain had to be decided, an issue not mentioned in the original papers. Implementation decisions will alter predicted performance, in the same way that cryptographic algorithm implementations are optimised, as observed in Section 4.2. For example, an entity might perform a memory lookup on an already verified signature rather than verifying the signature again. The database to prevent double spending is also often not considered. In an implementation, an extra expiry field or serial number in the payment instrument could remove the need for the broker to store and access an entire copy. New problems are likely to arise when a

scheme is implemented. To fully investigate our own micropayment scheme, we performed a full implementation, as described in Chapter 6.

The micropayment survey and performance analysis have provided a thorough understanding of micropayment properties and efficiency. In the next chapter we apply this to further advance the state-of-the-art by designing a first micropayment scheme allowing multi-party payments.

5 Multi-Party Micropayments

"The best way to predict the future is to invent it."
Alan Kay

5.1 Introduction

Any network communications that extend beyond the local administrative domain will involve entities belonging to different foreign networks. Typical examples include a long distance call in the PSTN, a mobile call placed to the fixed network, and a connection across the Internet. Ultimately, each entity through which the network traffic passes, or which provides part of the service, will be remunerated for their participation. Chapter 2 investigated how billing for such services operates in existing public networks; in the fixed PSTN, in mobile networks providing both circuit switched and packet switched services, and in the Internet. While telecommunications services have always been charged for, per-flow charging on the Internet is only emerging with the advent of QoS levels, IP telephony, and multimedia applications. Throughout Chapter 2, we highlighted the problems with existing billing techniques and we elaborate further on these in this chapter. The problems mainly arise from the fact that CDR billing originated for PSTN voice calls with only a small number of trusted network operators.

Already, this basic scenario has been augmented with millions of roaming users accessing both voice and data services through local mobile networks. Chapter 1 painted a picture showing how the whole infrastructure will change with the emergence of ubiquitous mobile roaming. In the future, anyone with local area wireless infrastructure can be a mobile NO and anyone with a service to offer can be a VASP. A random pedestrian might roam into a shopping centre, with its own Bluetooth [Haa00, Blu99] network. He could then connect through the high-speed local wireless link into a fixed network to place calls and use a variety of services. The billing methods of Chapter 2, with their implicit trust relationships, become drastically inadequate in such environments.

We can no longer assume a trust relationship between NOs, VASPs and roaming users. Instead, we propose that it is more secure and efficient to pay everyone involved in a call for their services as they are provided. Real-time payment eliminates the huge trust assumptions, security risks, and overheads of billing. Such payments for network services will be ongoing and for small amounts. Therefore there is a need for a real-time multi-party micropayment system, allowing users to pay every entity in a particular call route.

Such a payment scheme brings with it increased flexibility and opportunity. As long as the user holds a device capable of connecting to whatever network he finds himself in and provided he can electronically pay any party for services, the SPs can safely provide services to him. It no longer matters who the user is, what their credit rating might be, or where their home network is located.

Chapter 3 provided a detailed survey of all lightweight cryptographic techniques used in previous micropayment protocols. It was shown that multi-party payments have not yet been considered. Chapter 4

then took a close inspection of their performance and the efficiency of the underlying cryptographic constructs. Armed with a justified need, and an understanding of the efficiency trade-offs, we now design and present a *first multi-party micropayment protocol*. While it is well suited for use in mobile networks, it can also be applied to any services involving multiple parties in the fixed network, such as streaming video and IP telephony. Therefore, the protocol is de-coupled from specific network service protocols. Instead, we present a flexible protocol usable with any multi-party service model.

In Section 5.2 we introduce a model of our scheme describing the entities involved and how they interact. We then lay down the requirements of the protocol, making strict demands that the weaknesses of current billing protocols are eliminated. These requirements impose a much stricter security model than many payment systems; for example, we do not allow double spending whereas many schemes take the easier option of allowing it with post-fact detection. In Section 5.4, we describe how the user obtains prepaid monetary tokens. Sections 5.5, 5.6, and 5.7 detail the protocol that allows these payment tokens to be spent securely at multiple entities during a call. A scalable method of redeeming received payment is then described in Section 5.8. To ensure the security and robustness of our protocol we perform a security analysis in Section 5.9. From this, several potential weaknesses are identified and solutions to prevent them are incorporated into the protocol. In Section 5.10 the system performance is estimated and optimised. Finally, the system performance is compared with single-party micropayments, before concluding with the achievements of the protocol in Section 5.11. Earlier versions of the multi-party micropayment protocol appeared in our publications [PO99a, PO99b, PO99c].

5.2 System Model

A high level model of the system, its players and their interactions is shown in Figure 5-1. A *user* attaches to the network through an access network operator, either over a mobile wireless link or from a fixed terminal. As described in Chapter 1, many different types of wireless and fixed access networks will work alongside each other including GSM, UMTS, DECT, Bluetooth, HomeRF, wireless LAN, wireline LAN technologies, and PSTN links, amongst others. The scheme is network independent, and both personal mobility and terminal mobility are possible.

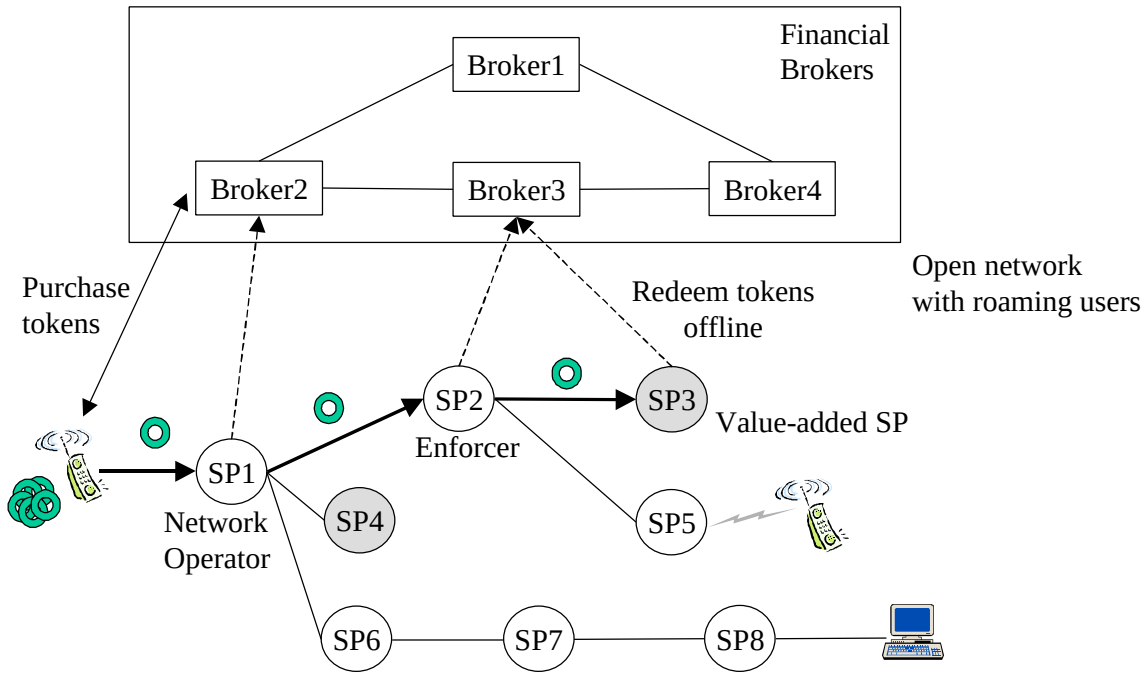


Figure 5-1 Multi-Party Payment Model

The user makes calls or sends packets through the access NO, for which he pays in real-time. Tariffs are dynamically set by each party at the start of a call. The connection may pass through one or more other network operators before reaching the destination user or VASP. A *Service Provider (SP)* is any entity who provides a service during that connection and includes both NOs and VASPs. The user releases a stream of micropayment tokens into the network to *pay all* the SPs as the call proceeds. The mobile sends a payment token to the local network operator who forwards a copy to all the downstream entities. The payment token is worth a different amount to each entity, and this amount is fixed at call setup. Thus SP1 could redeem each token for 2 cents while SP2 could redeem the same token for 3 cents. Tokens are based on hash chain constructions, which we showed to be highly efficient in Chapter 4.

Payment tokens are purchased by the user from one of several online brokers. The tokens are spent through a designated specific SP, called the *enforcer*, who prevents cheating by the user or the other SPs. Cheating by the enforcer itself will be detected after the fact. However, brokers will only bestow enforcer privileges upon those SPs that they trust, reducing the risk of fraud. After the call, payment tokens can be efficiently redeemed by each SP at their different chosen broker. One can envision a broker per area or region who will redeem for multiple SPs in that area. Only the final token received needs to be redeemed as the other tokens can be derived from this.

Unspent tokens can be spent on a different call to a different destination, but utilising the same enforcer to prevent double spending. If further calls are not made, tokens may later be refunded by the issuing broker. We present the protocol details of each step in the following sections.

5.3 Protocol Goals

Both existing and proposed billing systems have been outlined and a vision of the architecture of future mobile networks has been presented. We now provide the requirements of the multi-party payment solution. These requirements arise from the shortcomings of current CDR billing, demonstrated in Chapter 2, and address the problems these systems face in a large multi-SP environment.

1. Real-time payment to all parties from any location

A mobile user should be able to pay all parties involved in providing service in real-time, regardless of his current location and without need for online contact with a distant home location. This removes the need for subscriber billing and eliminates all of the associated costs imposed on network operators. Chapter 2 showed that existing mobile systems use strong authentication of users or equipment through a possibly distant home location. This is performed to allow billing, location management for incoming calls, and key management for ciphering. Once the user pays in real time, the home location does not need to be involved. In regard to ciphering, encryption key establishment can take place using the SP's public key. The need for subscription with a home operator can be completely removed if location management is treated as one of many services provided by VASPs.

2. Off-line payment verification

Any entity accepting payment should be able to efficiently verify its validity off-line, without the need to contact a third party. Each payee should be guaranteed to be able to redeem a valid token with a broker.

3. Remove user trust and accountability

The number of mobile users is far greater than the number of VASPs, which in turn are more numerous than network operators. Therefore, we propose that users should be the least trusted entities within the system, as shown in Figure 5-2. Existing credit-based mobile billing systems trust the user to pay his bill, based on initial strong identity verification, credit history checks, and strong on-line authentication at the start of a

session. Unlimited credit with post-fact punishment is too open to abuse in a large global system and has been a large source of fraud [Col00]. Extensive blacklists of stolen identities and equipment must be maintained to curb fraud and credit abuse. With so many mobile users, it is desirable to remove the need to trust them, and thereby minimise fraud.

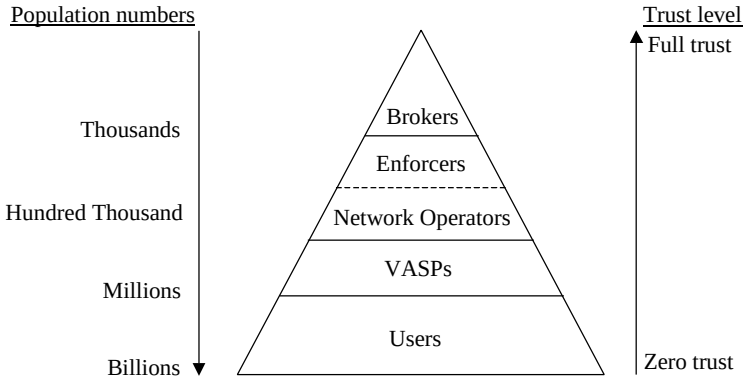


Figure 5-2 Trust Model for Multi-Party Network Payment

4. No user signatures and certificates

Mobile users are the least trusted entities in the system due to their numbers. While one can have some faith in a digitally signed document from a NO, a digital signature from a random roaming user, of which there can be billions, is of less value. Use of signatures implies the existence of a PKI [Lan98] capable of handling in excess of a billion certificates, assuming only one certificate per user. This is a huge task, especially considering that certificates will need to be revoked and the validity of a certificate checked by each party wishing to verify a user's digital signature. Protocols to verify certificates are usually online, such as the Online Certificate Status Protocol (OCSP) [MAMG+99]. The alternative is to circulate constantly updated blacklists. The scalability problems of both are well known [NN00, MR00, Riv98, NN98, Koc98, FL98, Bra99, FL99, Mye98, Mic96]. Solena and Harms [SH97] similarly argue that the choice of the user as the basic granularity for a global PKI remains the largest problem in deploying Internet security solutions. The largest PKI in current existence is only of the order of 10,000 users, several orders of magnitude below the number of ever-growing roaming users. In a global scenario, where the services of a very large number of independent entities can be used, the use of user digital signatures, with revocation checks, as a guarantee of payment will not be efficient or scalable. Also, current mobile devices, such as GSM SIMs, are not yet capable of generating public key digital signatures. In order to improve scalability and to remove the need to trust the user for payment, user digital signatures and certificates are not used within the system. This has the added advantage of affording the user more privacy.

5. Remove need for roaming agreements

Currently, operators must have roaming agreements with a foreign network in order for one of their subscribers to be able to roam into and place calls from that network. This results in a large number of bilateral roaming agreements. Such agreements should not be necessary to allow mobile roamers to make calls in whatever network they find themselves using.

6. Prevent inter-service provider fraud

Current CDR billing is based on trust relationships between NOs, who in turn must be trusted by users. Non-repudiation is not provided, allowing a network operator to forge CDRs. A mobile user can also deny making a call and hence refuse to pay the bill. With a large number of NOs and VASPs, the possibility of any fraud between these entities needs to be removed.

7. Dynamic charging

Both NOs and VASPs should be able to dynamically price their services on a per call or per request basis. Tariffs can then be adjusted depending on current network conditions, QoS provided, and user demand, amongst other factors. Current VASPs are restricted by the NOs static pricing model. Dynamic tariffs will allow a larger variety of services to be provided with different charging models. The need for dynamic charging for integrated Internet services is highlighted by Stiller et al. [SFPW98].

8. Payment flexibility

Current payphone solutions require the appropriate coins or prepaid cards to be able to pay the local network operator for the call. It should be possible to pay for a call using tokens specific to any entity that appears in the call route.

9. Identified payees

A payment token should only be redeemable by the intended payees. This is to prevent tokens being stolen by eavesdroppers and cheating entities.

10. Multiple brokers

A payment token should be redeemable and verifiable at any broker who trusts the issuing broker, without need to contact that issuing broker. This is unlike electronic cash systems where the coin must be returned to the issuer for verification. It should not be possible to obtain false value by spending the same payment token twice. Nor should it be possible for a single payee to redeem the same token with more than a single broker.

11. Portability of monetary value

It should be possible to optionally store the payment tokens on a small secure portable device, such as a smart card, and allow transfer from one device to another. This allows them to be used in any mobile or fixed terminal either owned by the user, rented, or shared. However, the security of the scheme should not require any special tamper-resistant hardware.

In summary, we wish to remove unnecessary trust from the system, reduce the online communications overhead of contacting a home location, eliminate fraud due to CDR tampering and falsification, provide fair dynamic charging, and allow real-time payment anywhere by anyone who holds valid payment tokens.

5.4 Payment Chain Purchase

A mobile user buys prepaid value, through their phone or terminal, from a third party broker using an existing macropayment system. The purpose of the broker is to aggregate micropayments between entities. To facilitate the purchase, we envisage that traffic to the broker will be accepted at no cost to the user. The payment chain purchase protocol is shown in Figure 5-3. The mobile user creates tokens by repeatedly applying a one-way hash function to a root value P_N to generate a *payment hash chain*. The user nominates any specific SP, called the *enforcer*, through which the tokens will be spent. This can be any NO or VASP in the call route. The chain has no monetary value until *committed to* by a broker. To obtain this commitment, the mobile user makes a macropayment to the broker, sending along the final hash P_0, the chain length N, the desired total value of the chain, and the identity of the enforcer through whom it must be spent, all encrypted with the broker's public key. It is assumed that the user has securely obtained and verified the broker's public key certificate beforehand. The root hash P_N, from which the rest of the chain can be generated, never leaves the mobile during the chain purchase phase. A multi-application smart card could be used to carry both the payment tokens and a macropayment instrument such as electronic cash or a credit card scheme, used to buy the payment chains.

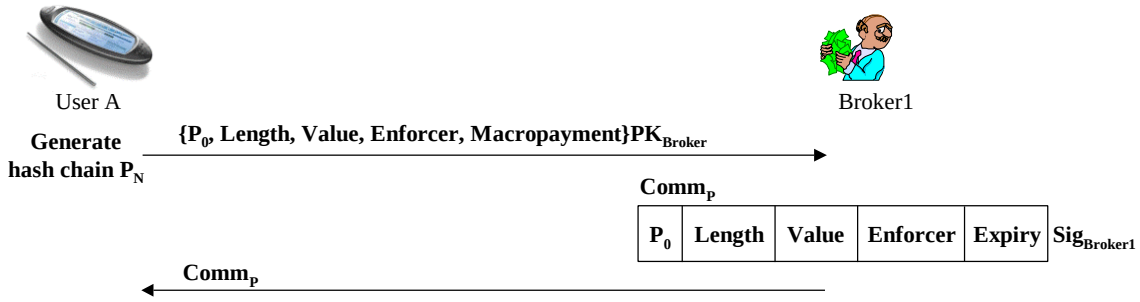


Figure 5-3 Payment Chain Purchase

The macropayment and public key encryption need only be used if the user is dealing with an unknown broker. Alternatively, if the user has an account, prepaid or otherwise, with a specific broker, a shared secret can be used both for authentication and integrity. An efficient message authentication code (MAC) function, such as the HMAC standard described in Appendix B, can provide the security:

$$\text{Account\_ID}, P_0, N, \text{Value}, \text{Enforcer}, \text{HMAC}_{\text{Secret}}(\text{Account\_ID}, P_0, N, \text{Value}, \text{Enforcer})$$

This is similar to secure withdrawal in the PayFairer scheme we proposed in Chapter 3.

The broker commits to the hash chain, or promises to honour its value, by digitally signing the *payment chain commitment* $Comm_P$, consisting of the chain details sent by the mobile user, as shown in Figure 5-3. The commitment shows that each *payment hash* from the chain represents pre-paid value, redeemable at the broker. The mobile user is given $Comm_P$, which is stored securely with the secret P_N. The value of a single payment hash is later fixed, on a per call basis, by the enforcer. This allows the *same hash value* to be used to pay all parties. By fixing the enforcer in the commitment, the mobile cannot spend payment hashes more than once by attempting to *double spend* at other providers.

A short expiry field is included in the commitment to limit the state that must be remembered by the broker to prevent double redeeming. Redemption must take place before expiry, after which user refunds on unspent value can be given. Similarly, if a payment chain is accidentally deleted, the unspent value may be re-claimed, provided the broker has a record of the chain owner. Monthly expiry fields can be efficiently represented using one or two bytes. If the mobile device is not capable of public key cryptography, a shared symmetric session key can be used to protect the macropayment details over the air interface, as with GSM devices. The signature on the commitment is verified by the payees at call setup, and will be rejected if invalid. This offers some security to a mobile device with no public-key signature verification capabilities.

The enforcer will prevent more than the total value of the chain being spent. Failure to do so will be detected by the broker when the hashes are redeemed. The chain length is included in the commitment so that the enforcer does not set the hash value to be such that it requires more hashes than are in the chain to be used to spend the total value.

5.5 Pricing Contract

To make a call, the mobile user must have a payment chain commitment for *any one* of the NOs or VASPs involved in the call. For example, a user might pay a VASP to provide him with voicemail facilities. He buys a payment chain to spend at the voicemail provider. Now, wherever he roams and whichever networks he

uses to access his voicemail, he can use that chain to pay not only the voicemail VASP but also all the network operators through which he is connecting.

If a call is being made to a party other than the voicemail provider, then the same payment chain should not be used. Instead, a different payment chain specific to one of the SPs involved in the call route will be employed. This prevents a geographically distant SP, not found in the route of the current call, being unnecessarily involved. A mobile user might typically carry several payment chains at once. Each chain would be specific to a service provider through whom or to whom he frequently places calls. As long as the SP is present in the route of the active call, the chain can be used to pay all SPs involved in that call.

If the enforcer for a new call is not the first local SP or the final destination SP then the call route needs to be known in advance in order to select an appropriate chain. We envision the local SP as being able to determine the call route, using network route queries if necessary. The user can inform the SP of the chains available and the SP can indicate one to use based on the call route. Alternatively, the user can be passed a list of possible enforcers along the route to the destination.

Figure 5-6 and Figure 5-7 show the same call being made but with payment chains for different enforcer SPs along the route. In Figure 5-6, the local network operator, SP1, is the enforcer. In Figure 5-7, the payment chain must be spent through SP3. For example this could be the voicemail provider from our example.

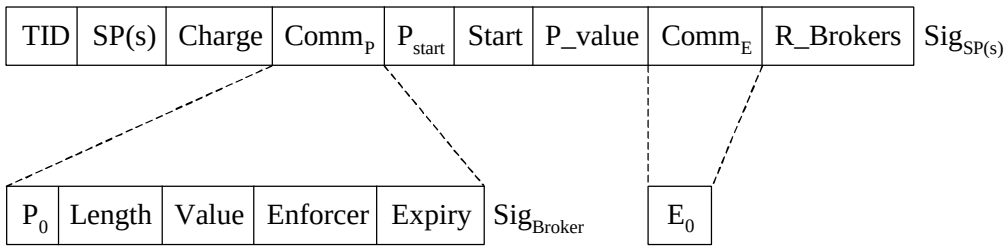


Figure 5-4 Contents of a Pricing Contract, Payment and Endorsement Commitments

To place a call, the user sends the call details, such as destination, call type, Quality of Service (QoS) requirements, and the payment chain commitment to the enforcer. A signed *pricing contract*, shown in Figure 5-4, is then generated by the SPs involved in the call. Its purpose is to allow verifiable dynamic tariffs; fix the starting hash in the payment chain; decide the value per payment hash for the call; create a record of the call; and link a single payment commitment to multiple SPs for the call. The fields in the pricing contract consist of:

- **TID**. Transaction identifier for the contract, partly generated by each SP. Each SP's part of the TID acts as a nonce guaranteeing freshness of the contract and preventing an old or partial contract being replayed to them.
- **SPs**. The identity of each NO and VASP involved in the call. When combined with the transaction ID, a unique identifier for the contract is obtained.
- **Charge**. Charging mechanism and individual tariff rate for each SP. The charging mechanism might be based on call duration, volume of data, or both.
- **$\mathbf{Comm_P}$**. Payment chain commitment, spendable through the enforcer.
- **$\mathbf{P_{Start}}$**. Starting payment hash from the payment chain for the call. This is the next unspent hash value, for reasons presented in Section 5.9.
- **Start**. Position of P_{Start} in the payment chain.

- **P_value**. The value per payment hash for the duration of the call, fixed by the enforcer. The value chosen depends on each SP's tariff rate and the charging method. If the broker cannot trust the enforcer to assign value to payment hashes, it can be fixed in the broker commitment, as with the first solution.
- **$Comm_E$**. An endorsement chain commitment, created and signed by the enforcer SP. This is used to prevent double spending of payment hashes, as discussed in Section 5.6.
- **Redeeming brokers**. Each SP fixes the broker through whom they will redeem payment hashes for this call. SPs within the same geographic area might use the same local broker to clear payments.

The charging mechanism and call tariff will vary according to the service requested, current network load, and time of day amongst other things. The pricing contract describes the tariff rate and charging mechanism, such as per second, per data unit, or QoS type, each SP will apply for providing their part of the service requested by the user. Since the pricing contract is signed by each SP, the mobile user can be assured that the tariffs are genuine and have not been inflated by the enforcer SP.

A problem arises when a payment hash may be redeemed by any party. Parties who were not involved in a call and who are not entitled to payment may try to redeem payment hashes. The pricing contract is used to identify those parties who may redeem payment hashes from a specific chain. An SP can only redeem a payment hash from a broker if he has a valid pricing contract that authorises him to do so.

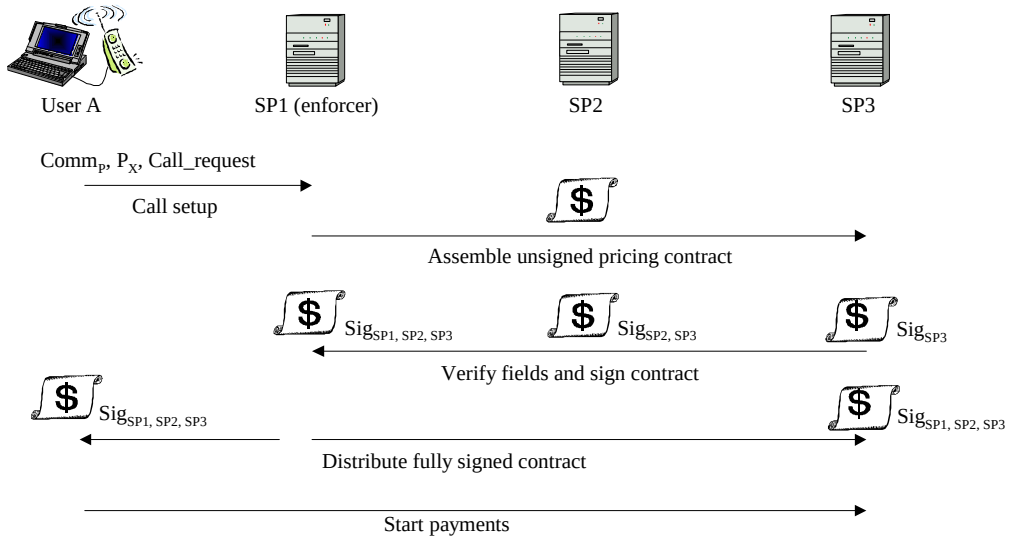


Figure 5-5 Constructing a Pricing Contract

The enforcer is responsible for ensuring that the pricing contract is constructed correctly using a three-way handshake protocol, shown in Figure 5-5. In step one, each SP adds their charging details to the contract. In step two, each SP digitally signs a hash of the fully assembled pricing contract, checking that their input has not been altered in any way. The SP signature does not include other SP contract signatures. The signatures later prove that each SP took part in the call and is due payment. The finished contract is forwarded to each SP in step three. There is no need for SPs to trust each other as each can independently verify the fully signed contract. The contract will only be accepted if each named SP in it has signed. The completed contract is presented to the user for agreement before the payment begins. From the charging information fields, the total call cost per charging unit is obtained. The user can verify the signatures to prove that each quote is genuine.

Figure 5-5 shows the enforcer signing the contract last, although this order is not critical. For example, when the pricing contract is being assembled for the scenario in Figure 5-7, the three-way handshake protocol may

occur in the same direction as before, but with the enforcer SP3 signing the contract first. While this allows an attacker to replace another SP's contract fields with incorrect values in step one and get the enforcer to sign these, such a contract will be rejected in step three as the genuine SP will not have signed tampered values. Furthermore, the enforcer knows which payment hashes have been spent, as they must pass through him, and he will not sign a contract with an already spent starting hash. Further security analysis is presented in Section 5.9.

If necessary, the public key certificates for the SPs are distributed to each other and the mobile in steps two and three. An SP's certificate is required in order to verify a digital signature from that SP. In a mobile device with limited computational capability, the signature verification can be omitted as long as the user is prepared to pay the price quoted. To detect over-charging, the validity of the pricing contract can be checked later on another device or with a broker. A new contract may be established mid-call to reflect any changes in tariff. For example, this might occur if a long call overlaps the switch from peak-rate charging to off peak rates. Similarly, if an inter-operator handover occurs during a call, a new pricing contract is established as part of the call setup with the new operator. A solution for coping with frequent handovers is presented in Chapter 7.

5.6 Enforcer Endorsement Chain

The enforcer, identified in the payment chain commitment, is given the role of preventing double spending of payment hashes. Since all payment hashes must pass through the enforcer he can keep a record of how much of the payment chain has already been spent.

After a call finishes, the mobile can re-use unspent hashes, the change, on another call which may pass through different networks to a different destination than the previous call. During the call, the enforcer will ensure that the payment hashes it and the other SPs receive, have not already been spent. However, without further protection mechanisms, colluding parties can fraudulently claim tokens to which they are not entitled. Consider the situation where a user spends payment hashes from a single chain during two different calls. In an attempt to commit fraud the user might then give payment hashes which were only spent during the second call to a SP who was present in the first call, but who was paid with hashes from a different part of the chain. Now both SPs from both calls have valid pricing contracts for the hashes, which they can redeem from the broker. Similarly SPs from each call could collude to swap payment hashes to gain value in this way. While the broker can detect this fraud when the same hash is later redeemed twice, he cannot be sure who committed the fraud and thus who should not get paid.

We introduce the concept of an *endorsement chain*, as a method to prevent SPs from redeeming parts of a payment chain to which they are not entitled. It is a hash chain created and committed to by the enforcer for each call. A hash chain commitment usually consists of the final hash, the chain length, and the identity of the chain creator, all signed. For an endorsement chain, the creator is the enforcer and the commitment would take the form:

$$Comm_{Endorse\_chain} = \{E_0, \text{Length}, \text{Enforcer}\}Sig_{Enforcer}$$

Every endorsement chain is unique to a call and is included in the pricing contract. However, because of this inclusion a number of optimisations can be made. The enforcer does not directly sign the endorsement chain alone because he later signs it as part of the pricing contract, as shown in Figure 5-4. In addition, the pricing contract already contains the identity of the enforcer, in the payment commitment, and so it may be omitted. Finally, the length of the chain is not required in the commitment because the chain will be made long enough so that the total remaining payment commitment value can be spent on the current call. No other

entity can benefit from knowing the length of the chain. Therefore the actual optimised endorsement chain commitment present in the pricing contract becomes:

$$Comm_{Endorse\_chain} = E_0$$

There is no value associated with an endorsement chain. Its sole purpose is to prevent double spending. Since endorsement chains contain no call-specific details, they can be generated in bulk beforehand. The anchor of the enforcer's new endorsement chain is included in each new pricing contract constructed. To make a payment, the mobile user releases a single payment hash. The enforcer applies a single hash function to verify it, and compares the result to the last received hash. If valid, the enforcer attaches a corresponding endorsement hash to each payment hash before forwarding it to the other SPs, as shown in Figure 5-6 and Figure 5-7. This endorsement hash indicates that the enforcer SP accepted the corresponding payment hash. An endorsement hash is specific to a call described by the pricing contract containing the endorsement and payment chain commitments. Unused endorsement hashes are securely deleted after the call by the enforcer.

Double spending by the enforcer SP cannot be prevented since he is entrusted with generating the endorsement hashes. However, if the enforcer does cheat, he will be detected after the fact when other SPs redeem the same payment hashes twice, as described in Section 5.9.5. Post-fact detection is acceptable for the enforcer because the broker will refuse to issue payment chains in the name of enforcers it does not trust.

5.7 Releasing Payments throughout a Service

Payment is *ongoing*, with the user releasing payment hashes at regular intervals. For a voice call this might be every ten seconds or for streaming video it might be for every 500KB. In return for a valid payment, the SPs continue to provide the service they agreed to in the pricing contract. If the user does not receive these services he can terminate the call by not releasing any more hashes. The total call cost per unit time, or per

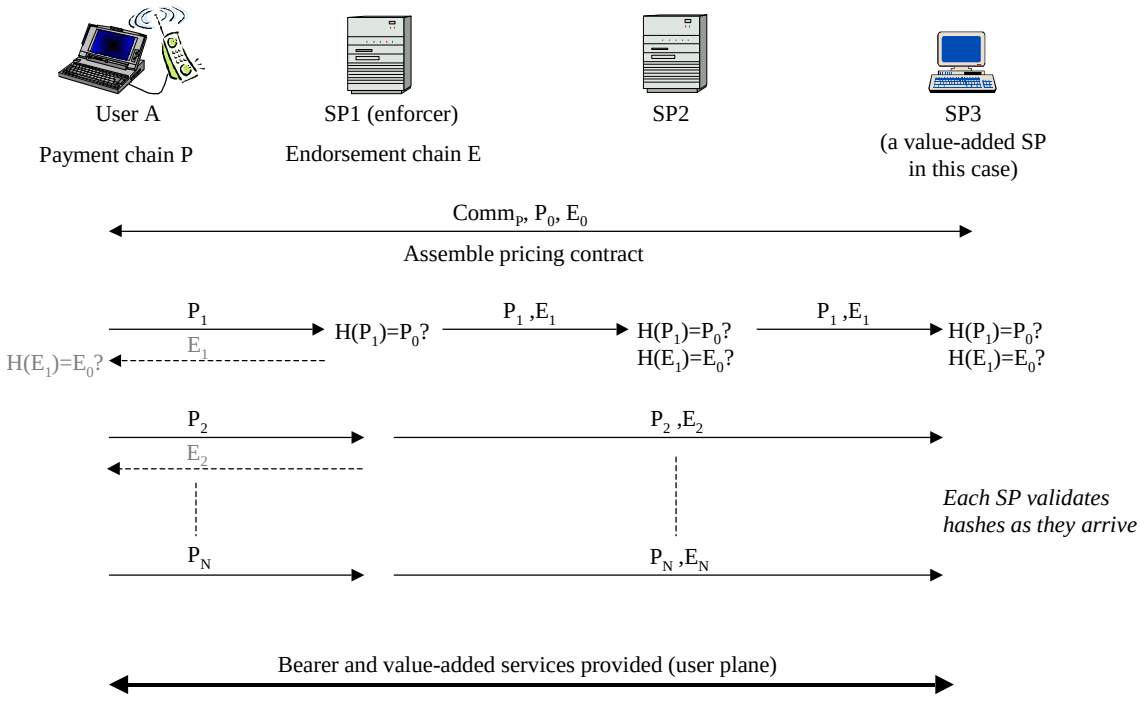


Figure 5-6 Mobile pays all SPs with same Payment Hash, with SP1 as Enforcer

data unit transmitted, is the sum of each SP's tariff rate in the pricing contract. The enforcer fixes each payment hash to be worth the total cost per charging unit. For example, in Figure 5-6, each SP might charge 1, 5 and 2 cents respectively per ten-second interval for a voice call. The enforcer assigns each payment hash to be worth 8 cents in the pricing contract.

Every ten seconds, the user releases a payment hash, in this case starting with P_1 from a new payment chain. The enforcer verifies that the payment is valid by performing one hash function on it to obtain the previous payment hash, in this case the starting hash P_0. The enforcer forwards the payment hash and his own endorsement hash to the other SPs. Each SP independently verifies both the payment hash and the endorsement hash. Since the hash function is one way, payment hashes cannot be forged, and knowledge of the payment hash is proof of payment. When the SPs redeem P_{40}, the 40^{th} payment hash, they will be paid 40, 200, and 80 cents respectively.

The enforcer endorsement hashes may also be sent to the user, as indicated by the dashed lines in Figure 5-6 and Figure 5-7. This allows the user to prove the call took place and prevents user value being stolen by the enforcer, as described in Section 5.9.5. Since enforcers are trusted by brokers and since all enforcer fraud is detectable, we conclude that such fraud is highly unlikely. Therefore, the endorsement hashes need not be sent to the user where wireless bandwidth is limited.

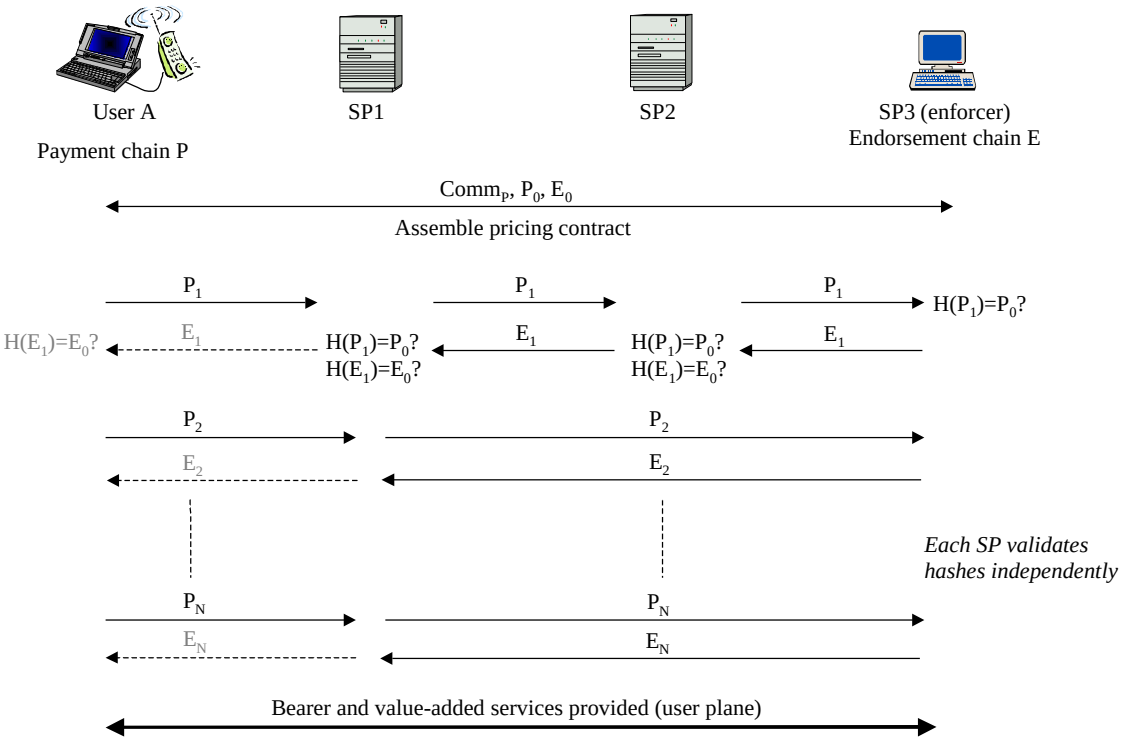


Figure 5-7 Mobile pays all SPs with same Payment Hash, with SP3 as Enforcer

Only the enforcer maintains state between calls. Every other SP only maintains *soft state* in the form of the last hashes received, which can be archived at the end of the call, to be redeemed later. The enforcer ensures that the total value of the chain is not exceeded. Once the total chain value has been spent, any unused remaining hashes have no value and are discarded. By including the commitments in the pricing contract, spent payment hashes, can be associated with a specific call. This creates a record of the call, providing much of the same information obtainable from a traditional CDR. When a pricing contract is associated with spent payment hashes the call duration or volume, call route and call destination can be obtained. Like regular CDRs this will allow it to be used for many secondary functions such as network planning, marketing data

collection, intrusion detection and law enforcement aspects. If desirable, and depending on the macropayment system used, the identity of a mobile user can be associated with a payment chain commitment when purchased at the broker. This would allow the caller's identity to also be associated with the call. If user anonymity is required, the payment chain must be purchased using an anonymous macropayment system, such as some forms of electronic cash.

5.8 Multiple Broker Clearing

Most micropayment and electronic cash schemes require that a payee redeem payment tokens directly from the issuer, so as to prevent double spending. In a multi-party payment, the requirement that all payees must go directly to the issuing broker is a serious restriction. For geographically disperse SPs this will introduce a communication overhead, even when performed off-line. To address this limitation we propose a network of brokers, where a payment chain may be redeemed at *any broker* agreed upon at the time of call setup. When the pricing contract is constructed, each SP fixes the *redeeming broker*, normally a local broker, with whom he is going to redeem the payment chain:

{SP1: BrokerA, SP2: BrokerB, … SPN: BrokerX}

No other broker can now redeem the part of the chain spent during the call, and hence double redeeming is prevented.

At the end of the day, each SP will redeem the highest spent payment hash from the call with their preferred broker. The broker will only accept a payment hash from an identified SP if a corresponding endorsement hash and pricing contract accompany it. In this way, double spending by the user and SPs other than the enforcer is prevented.

$$\text{Redeem} = \{\text{Pricing\_contract}, P_X, X, E_Y, Y\}\text{Sig}_{SP}$$

X and Y are the positions of the payment hash and endorsement hash respectively in the hash chains specified by the commitments in the pricing contract. Only the redeeming broker, fixed in the pricing contract by each SP, will redeem the hashes. The broker knows how much to pay each SP from the contents of the pricing contract. To verify the payment, the broker will perform *y* hash functions on each chain, where *y* is the number of payments made, to obtain the starting payment and endorsement hashes. The redeeming broker later clears payment chains in bulk with the issuing broker. Existing financial clearing networks could be enhanced to exchange these details. One can envision a broker per area or region who will redeem for multiple SPs in that area.

Fraud is only possible when a cheating enforcer signs multiple pricing contracts for the same part of a payment chain with different redeeming brokers for an SP. However, the enforcer fraud will be discovered when the SP redeems the same chain at more than one broker, and that enforcer can be blacklisted. Blacklisting of a small number of possible enforcers is far more scalable than blacklisting of the huge user population.

5.9 Security Analysis

Any payment system needs to be examined for possible attacks. In this section we show that our multi-party micropayment scheme meets our security requirements and is secure against attack. The security analysis exposed some vulnerabilities in the original protocol, for which we were then able to provide solutions. Before analysis, it was possible for an attacker to steal a payment hash and use it in any call. An enforcer

could steal any number of tokens from a user without the fraud being provable to a third party. Finally, an attacker could replay an eavesdropped payment commitment to get each SP to waste resources by signing a pricing contract that would never be used. Each problem is examined and fixed as part of the analysis. This results in a scheme which is more resilient and provides stronger security guarantees than the majority of micropayment systems in Chapter 3. Many of the attacks that we highlight are prevented by our scheme but present in others. We show that no outside attacker, user, or ordinary SP can obtain value to which they are not entitled. Enforcer fraud is shown always to be detectable. Methods which limit the effects of denial-of-service attacks are also presented. We make the assumptions that digital signatures cannot be forged, hash chains cannot be inverted, and an encrypted message requires the correct secret key to decrypt.

Our security analysis is based on writing a number of claims and proving each claim to be correct. It is thought that formal proofs remove the possibility of error or omission of attacks and highlight problems that would otherwise be overlooked. However, formal mathematical proofs are not utilised for the following reasons. In Chapter 3 when examining Lipton's probabilistic micropayments [LO98], we showed that computationally expensive techniques had to be added in order to formally prove its secureness. This rendered the scheme impractical showing the unsuitability of current formal techniques for micropayment analysis. In addition, the shortcomings of existing formal logic models, such as BAN logic [BAN90], when applied to payment systems has been highlighted [And97]. Constructs to model specific payment conditions do not exist in the model. There have been prominent examples in the cryptographic literature where payment systems which have been formally proved secure [Dam88], have later been shown to be insecure [PW91]. Finally, we point out that such formal analysis is complex and time-consuming and entire theses have been devoted to the formal proof of a single security protocol [Die97]. Efficient methods to formally analyse macropayment and micropayment schemes remain an open research problem.

5.9.1 Outside Attacker Fraud

An outside attacker is an entity connected to the network who is neither an SP nor a valid payment chain holder. An attack by an outsider, logically present between two valid entities on the network, is known as a *man-in-the-middle attack*.

Claim: *An attacker cannot obtain value during a payment chain purchase from a broker.*
The contents of the purchase request cannot be seen or altered as it is encrypted with the public key of the broker. The purchase response is the broker-signed payment commitment containing the anchor of the chain and can be obtained by an eavesdropper. Without knowledge of chain hashes, the commitment value cannot be spent. The secret chain values never leave the user until they are spent. An attacker can prevent messages reaching their destination, but a reliable transport protocol with re-transmission will handle this to a certain degree.

Claim: *An attacker cannot redeem value from a paid call, even if all payment messages are observed.*
A user will not release payment hashes until a valid enforcer-signed pricing contract has been received. The pricing contract specifies the SPs that can redeem hashes with a matching endorsement hash, and how much each hash is worth. Redeeming SPs must authenticate themselves, using a signature, to their chosen broker. Therefore, while an eavesdropper can obtain spent payment and endorsement hashes, they cannot be redeemed without breaking the authentication mechanism used with the broker. The pricing contract and endorsement hashes make payment hashes vendor-specific.

Claim: *An attacker cannot impersonate a value holder to obtain free service.*
The identity of a user who holds a valid payment chain is not needed for payment. SPs will only accept valid payment hashes for providing services and are not concerned with user identity. Therefore, without

knowledge of the secret unspent part of a payment chain, payments cannot be made and service cannot be obtained. However, there are circumstances under which an attacker can obtain a single unspent payment hash and spend it on a new call through the enforcer. This attack depends on which P_X hash value the enforcer will accept for a new pricing contract.

With a traditional hash chain scheme, a hash value can always be intercepted and spent at the vendor, with the user loss limited to that single hash value. The problem is potentially more serious in our multi-party scheme because the value of hashes is dynamically set by the enforcer. Therefore, an attacker will try to use a stolen hash on the most expensive service possible to obtain maximum value from the theft. We now show how the attack can occur and then provide a solution that prevents it.

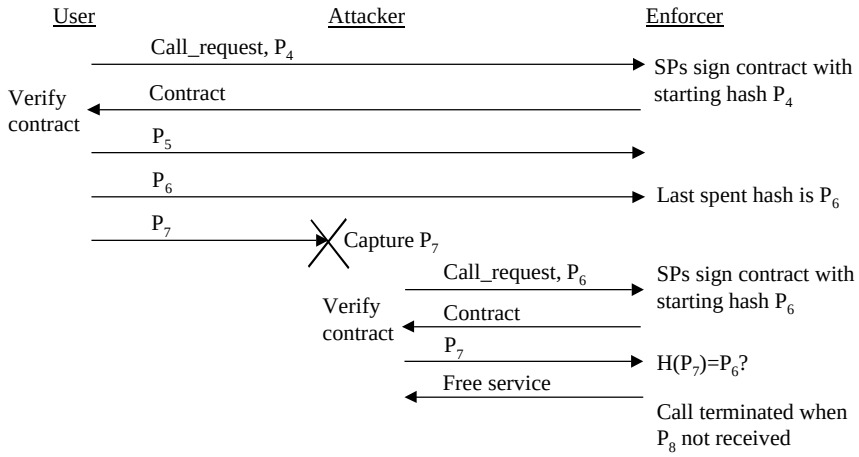


Figure 5-8 Active Attacker Stealing a Single Payment Hash

Each call request from the user contains a starting payment hash for that call. For the first call with a new chain this is P_0 but for later calls it will be a value higher up the chain. Assume that the starting payment hash is the last spent payment hash. For example, if P_6 was spent on the last call then this P_6 would be the starting hash, or temporary anchor, placed in the new pricing contract for the next call. The enforcer will know the last payment hash spent through him and will therefore accept P_6. However, it is possible that the user released P_7, and that the attacker prevented this from ever reaching the enforcer, as shown in Figure 5-8. After the service paid for with P_6 is used up, the enforcer terminates the call because he never received P_7. Since the user does not receive the service paid for with P_7, he certainly will not release P_8 or any further tokens. This leaves our attacker with knowledge of P_7 to P_0. He can now setup a new pricing contract with the enforcer using P_6 in the call request. To maximise use of his single stolen hash P_7, he will try to get a contract with as high a value as possible per payment hash, perhaps as high as €1.00 or more. He then releases P_7 as the first payment in this connection, as shown in Figure 5-8. The attacker cannot spend any further tokens because he does not have P_8.

We solve the problem, not by preventing the hashes from being captured, but by making them unspendable at the enforcer. The attack arises because the enforcer will assemble a new pricing contract based on an already publicly released hash value. In Figure 5-8, this publicly known hash value is P_6. If, instead, the enforcer requires a new unspent hash value for a new call, the attack is prevented, as illustrated in Figure 5-9.

In our example, the last payment received by the enforcer is P_6, even though the attacker has obtained P_7. Under our new requirement, the enforcer needs P_7 to set up a new pricing contract. While the attacker can provide P_7 with a new call request, he does not know P_8 and therefore cannot spend any value. Basically, we are now requiring a call request to be authenticated by including an unreleased payment hash.

Therefore, to make a new call, the user will always release the next unspent payment hash as part of the call request. Even if a pricing contract is constructed and no payments are made, a new payment hash will be required for the next call request. From the user's perspective, a payment hash is unspent if it has not been released or if it was released but no signed contract or endorsement hash was received. Hence, in Figure 5-9, the user would release P_7 as shown because it never reached the enforcer and hence no endorsement hash would have been returned. Returning endorsement hashes to the user is discussed in Section 5.9.4.

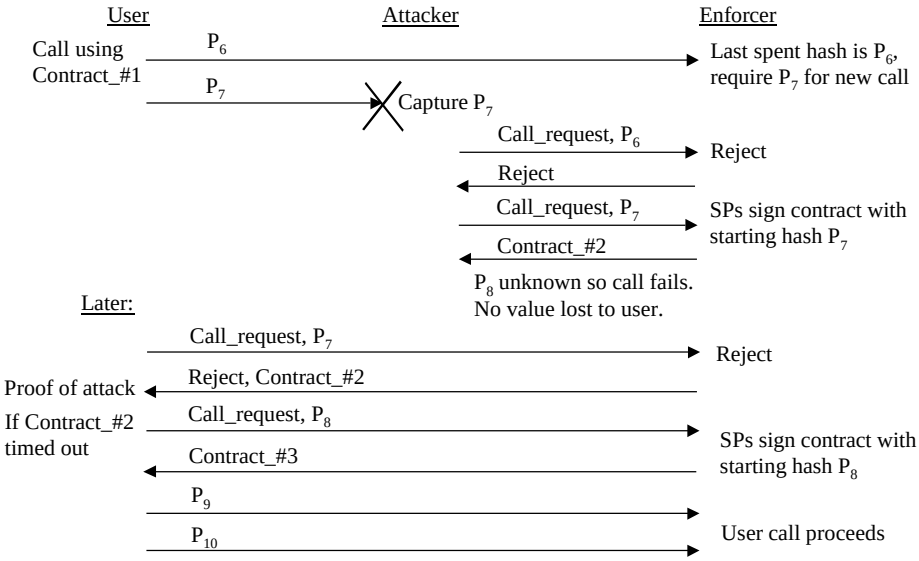


Figure 5-9 Preventing Robbery of Payment Hashes

If no attacker is present, P_7 will be accepted and a new pricing contract returned. However, since P_7 was already used by the attacker, it is rejected. The enforcer instead returns the last signed pricing contract, which is proof that an attacker is present. The attacker will not have been able to steal any value. The incident can be reported and a new chain obtained in exchange for the remaining value in the old one.

Alternatively, the user may release P_8 in a call request, as shown in Figure 5-9, provided that the attacker contract has *timed out.* If a first payment is not received shortly after a contract is established, its offer expires. This will prevent our attacker taking P_8 to use in his earlier contract.

Another effect of requiring a new token in a call request is that the token can be submitted by an active attacker as part of the previous call. Whether this is possible will depend on how connections are terminated, and the maximum time allowed between tokens before an active contract expires. In the unlikely event that such malicious attacks take place, strict contract timeouts may be set to prevent them.

An enforcer must only accept the next unspent hash for a new call request. If earlier hashes are accepted, this will become apparent when the pricing contracts are later redeemed. Our solution prevents an active attacker from being able to obtain any free service, other than eavesdropping on the user's connection, which is a passive attack. It requires only a single extra hash operation at the enforcer at call setup. The solution also reduces the effect of a denial-of-service attack, as discussed in Section 5.9.6.

Claim: *An attacker cannot impersonate a valid SP during a call.*
A valid SP holds a certificate, issued by a Certificate Authority (CA), and matching signature key. Even if an attacker does legitimately obtain such a certificate, he will be identified in the pricing contract and will be detected by other SPs who know their interconnect neighbours, or by the user. The attacker-signed pricing contract will also be proof of the fraud.

Claim: *No fraudulent value or free service can be obtained by colluding with any non-enforcer party.*
By colluding with a user, an attacker will have knowledge of a valid payment chain. Both user and attacker might attempt concurrently to spend the same chain through the enforcer. The enforcer implementation should ensure that payment hashes from a single chain are not spent twice, as this would allow a replay. In any case the enforcer prevents the total chain value being exceeded. No extra value can be obtained by colluding with an SP, as only identified SPs can redeem hashes as specified in the pricing contract.

5.9.2 User Fraud

Claim: *A user cannot spend more than the total value of a payment chain.*
The total value of the payment chain is specified in the broker-signed payment commitment. The signature prevents this value being altered. The value must be spent through the enforcer, who is fixed in the payment commitment. The chain is therefore a means of authenticating to a temporary account at the enforcer. The initial value of that account is the total chain value and, as payment hashes are spent, the enforcer decrements the account value. The enforcer will keep track of value spent and will prevent the total value being exceeded. The enforcer is liable for allowing user overspending.

Claim: *Payment hashes must be spent through the enforcer.*
Use of an enforcer hash chain in the pricing contract forces payment hashes to be spent through the enforcer. Without a valid endorsement hash matching a specific payment hash, the payment hash cannot be redeemed. The required endorsement hash is only released by the enforcer upon validation of the corresponding payment hash by him. Therefore, all payment hashes must first go through the enforcer. The enforcer chain is created by him, known only to him, and signed by him as part of the pricing contract.

Claim: *A user cannot double spend payment hashes.*
Payment hashes must be spent through the enforcer, as shown above. The enforcer will record the highest spent payment hash, in order to verify subsequent payments and in order to redeem hashes. Therefore, if an already spent payment hash is sent to the enforcer, he will detect it as not belonging to part of the chain above the highest received hash and will reject it. It is not in the interest of a user to re-use already released hashes because a network eavesdropper will already know these values. If the enforcer accepted old payment hashes, and decremented value from the remaining chain value, those replayed hashes could originate from an attacker rather than an actual user. For this reason payment hashes are only accepted once, to protect the user.

Claim: *A chargeable service cannot be obtained for free.*
An SP will only provide a premium service if they receive valid payment and endorsement hashes corresponding to a current pricing contract. Since the user cannot double spend or overspend, the only way to obtain service is by releasing valid payment hashes.

Claim: *Limited anonymity is provided.*
User anonymity to the broker depends on the macropayment used. No identity information needs to be included in a payment commitment, which allows the user to be anonymous to the enforcer, all SPs, and eavesdroppers. The anonymity is limited in that all transactions on the same chain are linkable from the pricing contracts. We wish to provide the user with privacy rather than full anonymity, due to the legal problems of totally anonymous money as described in Chapter 3. Where required, a broker can include identity information in the commitment, in a similar way to Millicent.

Claim: *Colluding with any party gains no additional value or services.*
No fraudulent value can be obtained or spent by co-operating with an outside attacker, as shown in the previous section. A user might collude with a non-enforcer SP in an attempt to obtain extra value. The user

can give all the chain secrets to any number of SPs. However, as shown above, a payment hash has no value to a non-enforcer SP without a corresponding enforcer hash and pricing contract. Therefore, extra payment hashes obtained from the user cannot be redeemed.

Not only can the tokens not be redeemed, but they cannot be spent through another SP. This would require a pricing contract with that SP named in it, and the matching enforcer hashes. The only way to get this would be to replay an old contract to the SP for which the enforcer hashes are already known. However, the TID included by the SP during contract signing acts as a nonce, guaranteeing freshness of the contract. The SP will not accept an old contract with an invalid nonce. Therefore, nothing can be gained.

A user might collude with the enforcer by revealing all of the payment chain. This will allow the enforcer to redeem the full chain value from the broker. The enforcer could allow more than the chain value to be spent at other SPs but, as we show in Section 5.9.5, this will be detected. The enforcer is liable for any overspending and so no extra value can be obtained.

5.9.3 Non-enforcer SP Fraud

A non-enforcer SP is a service provider which is not the named enforcer for the current payment chain.

Claim: *A non-enforcer SP cannot obtain more value than paid by a user.*
The value of a user payment hash to an SP is specified in the pricing contract, signed by the enforcer. The broker will use this value and the number of valid payment hashes, with endorsements, to calculate how much is owed to the SP. The broker must keep track of the highest payment hash redeemed in each contract and which SPs have redeemed it. The recorded state is limited by an expiry field in the payment commitment. To increase the value per token requires the pricing contract to be altered, which is not possible without forging signatures. To obtain extra hashes requires the co-operation of both the user, for payment, and the enforcer for endorsement. The enforcer will not aid SP cheating as the enforcer becomes liable for losses.

Claim: *An SP cannot obtain value belonging to another SP.*
All SPs redeem the same tokens, with value calculated from the contract. To obtain another's value requires authenticating as that other SP to their nominated broker. Again this requires a forged digital signature.

Claim: *Pricing contracts cannot be replayed at call setup without detection.*
An SP might try to replay an old pricing contract, or a partially signed contract, to double spend collected tokens. Every SP does not need to keep a record of previous contracts to check against at call setup, as this would be too cumbersome. Instead, the SP TID placed in the contract in step one of contract assembly forms a nonce. If an old contract is replayed in step two, it will have a different nonce, and can be rejected. This is the reason for the three step contract handshake.

In addition, a valid contract requires the enforcer's signature, which locks the SP identities into the contract. An SP not present in the enforcer-signed contract will not accept it, as it means it cannot redeem the tokens. Replaying partial contracts, not yet signed by the enforcer, will be detected by the enforcer due to an old nonce or an SP signature on incorrect values.

Claim: *Extra value cannot be obtained by collusion.*
The cases of colluding with a user and an attacker have already been shown to have no gain. Collusion with another SP in a single call similarly achieves nothing as the pricing contract locks the value redeemable by each SP. A new endorsement chain is used in each pricing contract. This prevents payment hashes from one call being redeemed by an SP in another call using the same payment chain. An endorsement hash links a

payment hash to a specific contract and therefore to the SPs listed in that contract. Therefore, endorsement hashes prevent SP collusion. The enforcer is liable for chain overspending and, therefore, it is not in his interest to help SPs commit fraud.

5.9.4 Enforcer Fraud

Claim: *Payment chain overspending by the enforcer can be proved to an independent third party.*
The broker records the total amount redeemed against a payment chain. When more than the total value is spent, it will be detected by the broker when redeemed. The redeemed pricing contracts are proof of the overspending, and show exactly how much each party redeemed. The overspending fraud occurred with the enforcer's consent, since earlier claims showed that fraud by any entities other than the enforcer was not possible.

The enforcer can cheat in a number of ways. User payment hashes can be obtained in a normal call with a valid pricing contract, and can then be double spent by the enforcer. For example, the enforcer can create a second pricing contract, paying himself more than he is entitled to. Alternatively, he can double spend at other SPs with another pricing contract, and hence obtain free service for himself. The enforcer may also allow fraud to occur from which he does not directly benefit, but through which he may profit by having an agreement with the party he allows to overspend. Such cases are where the enforcer allows the user to overspend, or where he releases extra endorsement hashes allowing SPs to collude. The redeemed endorsement hashes prove the enforcer's guilt in such scenarios.

The worst the enforcer can do is redeem the total chain value for himself, and use the chain to provide many false payments to any number of SPs. However, in each case the fraud will be detected as having been committed by the enforcer and can be proved using the enforcer signatures and endorsement hashes. In such cases enforcer privileges and certificates can be revoked and legal action taken. In addition, a broker will only bestow enforcer privileges on trusted SPs, making enforcer cheating even more unlikely.

Claim: *Unprovable enforcer fraud is limited to stealing a single payment hash.*
We first outline how an enforcer can steal chain value from a user, although the user will detect this. We then show how only a single payment hash can be taken without being able to prove the fraud to a third party.

A payment commitment represents a temporary account at the enforcer. The payment chain hashes authenticate the user to that account and prove to a third party that he agreed to pay. The enforcer can obtain valid payment hashes from a call with a valid pricing contract. The enforcer can then construct a fraudulent pricing contract with the spent payment hashes. The cheating enforcer will set the value per hash to be high enough that he can redeem the total chain value for himself using the captured payment hashes.

If the original payment contract is never redeemed, then the broker has no proof of fraud. Circumstances under which a contract is not redeemed would be if the enforcer was the only SP in the call, or if the other SPs collude with the enforcer. The fraud relies on not being able to prove that the original call took place. However, by requiring the enforcer endorsement hash also to be sent to the user, as shown in Figure 5-6 and Figure 5-7, the user will hold proof that the corresponding payment hash was spent with the original pricing contract. The user will not release further payment hashes until the endorsement hash is received.

If the enforcer constructs a bogus second contract to steal value, the user can present the original contract and endorsement hashes as proof of enforcer double spending. The worst the enforcer can now do is accept a payment hash during a call, and not release the endorsement hash. Alternatively, he can accept a payment hash for call setup, never sign a new pricing contract, and redeem the token as part of the previous call. In

both cases, enforcer cheating has been limited to one payment hash. This is because each payment hash released is acknowledged by the enforcer either with a new signed contract or an endorsement hash. Unlike delayed CDR billing, the user becomes immediately aware of enforcer fraud on the next call attempt. A user will not use the services of an enforcer who defrauds him, and the enforcer will lose reputation and business. A malicious user, in an attempt to defame an enforcer, might deny having received an endorsement hash and falsely claim that the payment token was meant for a new call but was subverted by the enforcer. However, such an accusation, whether holding any truth or not, cannot be proved.

Claim: *Collusion can make enforcer fraud easier but does not make it less detectable or provide any additional fraud capabilities that the enforcer cannot perform alone.*
Collusion with an outsider or another enforcer gains nothing for the chain enforcer, as they do not have the payment chain or take part in the call. Colluding with a user allows the entire payment chain to be known to the enforcer. This may make it easier to double spend payment hashes at other SPs to obtain free service. However, the fraud will still be detected when contracts are redeemed. Unprovable enforcer fraud has been limited to stealing one payment hash. The original contract used to steal this hash must not be redeemed if the enforcer is to escape provable detection. A colluding SP can be allowed in the original contract if they do not redeem it, thereby helping the enforcer. However, in each case, the enforcer can perform the same fraud alone, as examined in previous claims. Enforcer fraud is highly unlikely as it will always be detected. Our scheme is far more secure than other micropayment schemes, many of which we showed to allow unlimited user and vendor fraud.

5.9.5 Broker Fraud

Claim: *The amount owed to each SP by the broker can be proved to an independent third party.*
SPs redeem value from the broker with the pricing contract and the highest payment and endorsement hashes received. The value of each hash to the SP is stated in the contract. Therefore, any party can verify the amount owed to a specific SP. Broker payment to an SP will be an auditable business-to-business electronic payment.

5.9.6 Denial-of-Service Attacks

A *denial-of-service attack* attempts to deprive legitimate users of system access. Such attacks often flood the system with messages which must be processed as potentially valid requests, thereby preventing valid calls being accepted. In our multi-party scheme, the most computationally expensive stage is the call setup where each SP signs the pricing contract. A denial-of-service attack might repeatedly send call setup requests, getting each SP to waste resources signing a new pricing contract that is then never used. The effect of such an attack on resource usage needs to be limited or alternatively the cost of performing the attack needs to be made prohibitively large. While such computational flooding attacks may prevent a valid call being accepted, another approach is directly to attack the payment instrument. This might be done by invalidating unspent value or blocking issued change. In this section, we show how the protocol is made robust against these two serious attacks, thereby greatly reducing their possible impact.

If the call setup request only requires a valid payment commitment, a denial-of-service attack can be easily launched. In Section 5.9.1, we showed that an outsider can eavesdrop the payment commitment, either during withdrawal or call setup. The eavesdropped commitment can then be replayed to the enforcer causing resources to be consumed as the contract is signed by all SPs. The call will not proceed because the attacker does not hold valid payment hashes from the chain. A simple countermeasure would be for the enforcer to temporarily refuse to accept a specific commitment after this occurs a number of times in succession within a short time frame. However, this then provides an attacker with means quickly to suspend a user’s ability to

spend a chain, which is an even worse attack against the system. Such attacks against the user are discussed in a later paragraph.

Claim: *Requiring proof of payment chain ownership limits the effect of a denial-of-service attack.*
In Section 5.9.1, we showed that an unspent payment hash should accompany each new call request. The purpose of this was to prevent an attacker stealing a single token. In fact, this requirement also efficiently limits the effect of a denial-of-service attack on call setup, as shown in Figure 5-10. The payment hash in the call request proves the presence of the chain owner before the call is setup. Between calls there is always at least one payment hash used for authentication rather than payment.

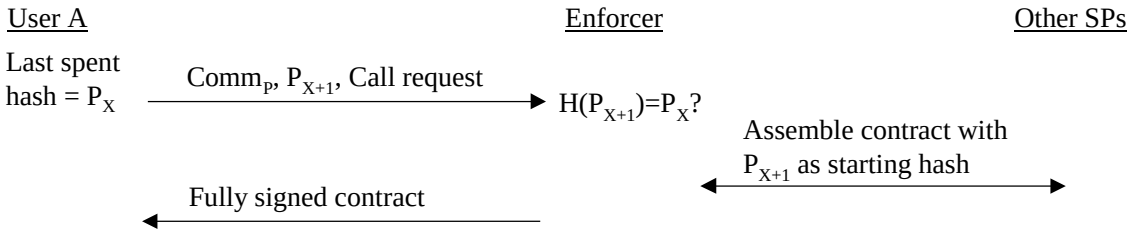


Figure 5-10 Commitment Authentication to Reduce Effects of Denial-of-Service

Unless a valid unspent token is presented, no new pricing contract is assembled. Section 5.9.1 showed that it is possible for an attacker to obtain a single unspent token and use it to assemble a new contract. However, when the user later attempts a new call, this attack will be discovered before releasing any more tokens. Therefore, we have limited the denial-of-service attack on call setup to a single bogus contract assembly instead of possibly infinite contracts. An attack using invalid tokens will never get past the enforcer. This is a vast efficiency improvement over allowing all unauthenticated call requests to assemble a fully signed contract.

The enforcer is still limited to stealing a single valid hash because, for each call attempt, the payment hash used becomes the starting hash in the call pricing contract. If the enforcer does not produce such a contract, the user will know that something is astray and not release further payment hashes for authentication. In this way, the enforcer, or an eavesdropper, is prevented from stealing a number of payment hashes.

Claim: *By invalidating a single payment hash an attacker cannot prevent the remainder of a payment chain being spent.*
An alternative attack to computational flooding is to block or invalidate unspent monetary value. The obvious attack here is to intercept the payment instrument and stop it reaching its destination. However, this requires the user's communications to be constantly monitored and blocked at the appropriate time. A more successful attack would be to steal change from a transaction or perform actions which cause the payee to refuse all further payments from the user. For example, it is possible to block change in micropayment schemes such as Millicent, SubScrip, NetPay, and Mao's MicroCash.

In our scheme, we would like to ensure that an attacker cannot prevent the user spending remaining value in a payment chain. In Section 5.9.1, we showed how an attacker can obtain an unspent hash and use it to setup a new contract. Even though he cannot steal any value, the user's valid call request with the same hash will later be refused by the enforcer. The last signed contract is returned to the user with the rejection, as shown in Figure 5-9. The contract is proof that the hash has been used, without the user's consent, and authenticates the rejection as being from the enforcer.

The user can now release the next unspent hash to authenticate himself as the chain owner and establish a new contract. As discussed in Section 5.9.1, care must be taken to ensure that the time in which a hash can be accepted for the attacker contract has timed out. If necessary, this timeout value can be placed within the pricing contract. This not only prevents an attacker stealing that hash, but also allows the user to continue without being locked out of using the chain. Therefore, our solution is more robust than other prepaid micropayment schemes which issue change, as an attacker cannot deny the user access to remaining unspent value.

By adapting the payment protocol, we have reduced the ease and effects of the most serious denial-of-service attacks. Other attacks include sending false hash values or blocking communications altogether. While these will disrupt a call, they do not consume as much computational resources as the original attack on call setup. No software-only approach can prevent such attacks from occurring on open networks.

5.10 Performance Estimates and Optimisations

The theoretical performance of the system in terms of computation, storage, and communications is now examined. We apply the same performance analysis techniques as when evaluating the micropayment protocols in Chapter 4. The performance estimates of the multi-party micropayment protocol are presented in Table 5-1. The assumptions and calculations used to obtain these figures are given in Appendix C. The figures are discussed and, based on this, optimisations are made to improve performance. To facilitate comparison, the optimised estimates are summarised in the same table. Comparisons are also made with CDR billing and other micropayment protocols.

	Original			**With optimizations**		
Computation (#hashes)	User	Enforcer	SP	User	Enforcer	SP
First payment	714	3625	3624	238	3149	238
Average payment	49.5	50.5	2	49.5	5.5	2
Storage (bytes)						
Certificate storage	620	748	1058	620	748	438
Payment inst	179	838	677	179	445	421
Communications (bytes)	U->Enf	Enf->SP	SP->SP	U->Enf	Enf->SP	SP->SP
First payment	1311	1705	1593	435	843	1169
#messages (first payment)	2	4	4	2	4	4
Average payment.	20	40	40	20	40	40
# messages (avg. payment)	1	1	1	1	1	1

Table 5-1 Performance Estimates for Multi-Party Micropayment Protocol

From Table 5-1 we see that the storage costs of the scheme are very acceptable. The user stores under 1Kbytes in total, which is suitable for devices with limited storage. The flexibility of spending at any party through the enforcer makes it likely that only a few chains with well-placed enforcers will be required. This is unlike other vendor-specific payment instruments. Only the broker and enforcer certificates are kept, unlike other SP certificates which can be discarded after use. The storage of payment material after a call is almost 700 bytes for an SP. The majority of this space is occupied by the three SP signatures. Nevertheless, a contract typically will not be stored for longer than a day, unlike traditional CDRs, which are kept for months.

At first, the communications costs appear a little high at approximately 1.5K between each pair of entities. The number of messages is either two or four, which means the individual message sizes are smaller. The majority of communication is due to passing around not only the signatures but also the certificates necessary to verify them. Each SP signature will require the matching SP certificate. To reduce this overhead,

certificate thumbprints, indicating which certificates an entity already holds, could be exchanged, as in the SET protocol [MV97]. This prevents a certificate being unnecessarily sent to an entity if it already holds it.

In addition, the three-way handshake message results in the same material being passed three times between entities, inflating the communication used. If bandwidth was a problem, contract contents could be cached rather than sending them multiple times. However, we use the network as an efficient memory storage. The majority of communications is among the SPs in the fixed network rather than across a wireless link to the user.

Our analysis has concentrated only on the overhead of the first payment. Ongoing payments during a call are very efficient, sending only 20 bytes over the wireless link, and 40 bytes across the fixed network. Only one hash is required to verify the payment token by all parties, and a second hash by the non-enforcer SPs to verify the endorsement token. The greater the number of payments made, the less the total call overhead becomes. Again, the first payment at call setup is more computationally expensive, requiring a number of signatures and verifications, as detailed in Appendix C. We now consider some optimisations to reduce the overhead at call setup.

Optimisation: *Each entity need not verify all the non-enforcer SP signatures on the pricing contract.*
We show that for each ordinary SP, enforcer, and user, it is not necessary to verify SP contract signatures, and the implications of this. In step three of contract signing, each party verifies the fully signed contract, including ordinary SP signatures. An ordinary SP needs to be assured that the fields it added to the contract remain unchanged and that it has been signed by the enforcer. The SP will not be paid correctly if either of these conditions is not met.

For a call with *N* non-enforcer SPs the *(N-1)* signatures of other SPs are also verified by each ordinary SP. However, it is not necessary for the SP to know before a call who the other SPs involved are. They will still receive correct payment regardless of who the other entities are. Therefore, an ordinary SP need only verify the enforcer signature. If, for some reason, such as checking a competitor's prices, or for traffic analysis, the SP does want to verify the presence of the other SPs then this can be done offline after the call.

Similarly, the enforcer is concerned with getting paid and preventing the user from overspending. It is not relevant to him at the time of the call, who the other SPs are or what their tariffs are, provided that the user is prepared to pay the sum of the individual tariffs.

An SP signature authenticates their tariff and proves their presence to a user. If the user is not concerned with authenticating each SP and is happy to pay the presented tariff prices, signatures other than that of the enforcer need not be checked. If, after releasing the first payment token, the promised service is not received, the user will not release any further tokens.

Each SP checks the integrity of its own charging field and signature when verifying the contract, and will reject the call if not correct. Therefore, the consequences of all entities not verifying the SP signatures is only that an attacker can include false SP identities and tariffs. However, value cannot be redeemed without authenticating to the broker, who can also verify the contract at that time. Therefore, such an impersonation attack can only damage reputations rather than obtain value.

If such malicious attacks become a problem we suggest probabilistic verification of SP signatures. An entity will decide, based on a certain probability, whether to verify the other SP signatures or not. The probability of performing a verification should be directly proportional to the frequency of discovering invalid signatures

while checking old contracts offline. Another approach is to place the signature verification overhead entirely on the enforcer. The enforcer should not sign the pricing contract unless he has successfully verified all signatures on it, thereby removing the need for anyone else to check them. In either case, we have reduced the number of signature verifications by users and ordinary SPs to only one, the enforcer signature verification, for an entire call.

Optimisation: *Non-enforcer SP signatures can be eliminated.*
The previous optimisation showed that it is not essential to verify each SP signature at call setup. This leads us to consider if SP signatures are necessary at all if no party is verifying them. The broker still verifies SP signatures on the contract when that SP redeems tokens. The signature guarantees that the SP was part of the call and is due payment. An SP must authenticate itself to the broker, thereby preventing an unidentified eavesdropper stealing value. However, the authentication at redemption need not rely on an SP contract signature, as presence of the SP identity in the enforcer-signed contract is adequate.

The consequence of removing the SP signature is that proof of that entities participation and tariffs in a call is removed. However, if an authenticated SP redeems a contract then it is agreeing that its fields in the contract are correct and that it took part. Therefore, the SP contract signature can be replaced by an offline SP signature during redeeming. After all, SPs have performed authenticated redemption, the broker has the same proof of presence as with SP signatures on the contract.

Removal of SP signatures affords an attacker the same impersonation opportunities as when the signatures are not verified. If the attacker alters contract entries belonging to other SPs during contract assembly these will be detected as before and the call rejected. In addition, SPs should know of their neighbouring interconnected entities and, if a false SP identity is included by their neighbour, they can also reject the call. While an attacker can impersonate another entity during the call he cannot authenticate as that entity to the broker to redeem value. However, it will appear to the enforcer that the value released to the fake SP has been spent. If such malicious attacks become problematic, SP signatures can be re-introduced.

Optimisation: *Non-enforcer SPs need not verify the payment commitment.*
The enforcer will verify the broker signature on the payment commitment. If it is valid, the enforcer will include it in the pricing contract which he then signs. Therefore, the enforcer signature on the pricing contract is evidence that the enforcer checked and accepted the commitment. However, the enforcer can include an invalid commitment, although this can be detected later when the contract is redeemed. For this reason, it is not necessary for any other SP to verify the broker signature as long as they can accept post-fact detection of enforcer cheating.

Optimisation: *The enforcer only needs to verify the payment commitment once for all calls.*
The user sends the commitment to the enforcer for each new call, along with an unspent hash, as proof of ownership of the value spendable through that enforcer. The enforcer keeps track of the total amount spent and last received payment hash, associated with the payment commitment.

If the enforcer keeps a copy of the commitment, the signature need not be verified for each new call. In fact, the user only needs to send the next unspent payment hash and an identifier, such as P_0, which indicates to which commitment, stored at the enforcer, it belongs instead of the entire commitment. Alternatively, the enforcer stores the commitment identifier, P_0, and the user sends the entire commitment each time. In this case, the signature will need to be verified before including it in the contract. The optimal choice will depend on whether a database lookup and comparison of the entire commitment is more efficient than re-verifying the broker signature. In Chapter 4 we measured 220 signature verifications in Java, while the micropayment

database discussion in Appendix B suggests that only 100 database lookups are possible per second on a commodity computer. It may therefore be more efficient to re-verify the commitment for each new call.

Optimisation: *When sending the pricing contract to the user, the payment commitment may be omitted.*
The user already holds the broker-signed commitment. If the user is setting up multiple calls at once, the returned contract can be matched with the appropriate commitment, for signature verification, from the starting payment hash. The commitment could also be omitted from steps two and three of contract assembly. However, if an SP is setting up multiple simultaneous calls for different users they may not want to keep each commitment in memory and match it up with the appropriate incoming message. From a computational viewpoint it is more efficient for a heavily loaded SP to receive the entire contract from the network.

Optimisation: *Enforcer hashing can be reduced by decreasing the distance between cache points in the endorsement chain.*
It is likely that a user will only spend a fraction of a payment chain on a single call. Therefore the endorsement chain can be kept much shorter than the payment chain length. For an equal number of cache points, the distance between cached chain values in an endorsement chain will be much less than those for a user chain. We have assumed every 100 values in a user chain are cached. Assuming only one tenth of a payment chain is spent per call, the enforcer can cache every 10 values in the endorsement chain. This reduces the average number of enforcer hashes to obtain the next endorsement hash to (m-1)/2 = (10-1)/2 = 4.5.

We applied the above optimisations to the original multi-party protocol and re-calculated the performance estimates, as explained in Appendix C. The improved results are presented alongside the original predictions in Table 5-1.

5.10.1 Performance Improvements after Optimisation

The improvements obtained over the original protocol are now discussed. The biggest saving is in SP computation during the first payment, which is reduced to 6.6% of its original value. This is very important when a number of SPs are involved in a call, especially if they are present in the high-traffic core network. This huge improvement allows each SP to process a far greater number of simultaneous calls.

The user computation is improved three-fold, reaching a level where multiple calls can be set up simultaneously, even from the smallest device, based on performance figures in Chapter 4. The enforcer computation is also reduced, with the enforcer contract signature still requiring the bulk. We have greatly reduced the enforcer computation during a call to under six hashes per payment. Payment processing for every other SP remains extremely efficient at two hashes. Therefore, the work performed by all SPs is kept to a minimum. The average number of hashes per payment remains at under 50 for the user, due to long chains and assumed limited caching. However, we already showed in Chapter 4 that even the most computationally limited device can perform thousands of hashes a second, proving that the ongoing payment cost is highly efficient.

Elimination of SP signatures has resulted in a much smaller contract size. This yields a very reasonable payment material storage size of under 500 bytes at all parties. Indeed, this is less than some modern CDRs which record every service change and action during a call.

In addition to the smaller contract size, the removal of the need to exchange SP certificates also results in a large reduction in communications traffic for the first payment. The number of messages exchanged remains the same although each message is now smaller. Communications between the user and enforcer is one third

of the original protocol, less than 450 bytes. This is very acceptable for a wireless link. Not returning a copy of the payment commitment made this improvement possible. Enforcer-SP traffic is halved and inter-SP traffic is reduced by 500 bytes. Message sizes for the majority of payments remains very small at 20 bytes over the radio link, and 40 bytes in the fixed network.

5.10.2 Comparison with Billing and Single-Party Micropayments

The performance of our multi-party scheme is now compared with CDR billing and single-party micropayments. The comparison is purely on performance and does not consider all the security advantages and flexibility that our protocol offers over traditional methods.

Chapter 2 showed that CDR creation simply involves logging switch state information to a file. The introduction of a payment system will obviously introduce an extra computational overhead above simply writing to a file. However, in Chapter 2, we also showed how any change in call conditions, be they in radio bandwidth, provided QoS or otherwise, required that change to be recorded in a CDR to allow the correct amount to be billed. Real-time payment is more efficient – as a service is dynamically degraded or enhanced the user just releases fewer or more payment tokens to pay for the current service level, as specified by the pricing contract.

In addition, we showed how the size and variety of CDRs is becoming greater and greater in order to record every event and state change during a call. With real-time payment, only the payment material needs to be stored, and this is likely to be smaller than many CDRs. Payment material is also redeemed daily while CDRs must be kept at least until the end of the billing cycle. Therefore, we claim that for similar daily storage costs, reduced long-term storage costs, and some additional computation, our multi-party protocol eliminates many of the current and future problems of existing billing systems.

Our micropayment survey in Chapter 3 showed that there has been no other attempt to create a multi-party micropayment protocol. One possible option is to use a single-party micropayment scheme to pay each entity individually, requiring N separate micropayments to N SPs. Therefore, when making comparisons with other schemes, we also consider their performance when used to independently pay 3 SPs. However, multiple independent payments are not secure for several schemes since a competing intermediary SP can steal value. Such is the case with vendor-independent tokens, such as MicroMint or Wheeler's bets, and change issuing schemes such as SubScrip or Millicent. We do not make comparisons with these schemes or with online or public key based micropayments, such as Tang's scheme, NetBill, or Mini-Pay, as our survey showed them to be far more expensive than other schemes and unsuitable for frequent payments.

With single-party micropayments, a new payment instrument is required for each new call, while, with our scheme, the same chain can be used at all SPs, even across multiple calls. A single-party prepaid hash chain scheme will require three different chains to be purchased and stored for each call. In a credit scheme, the user will have to generate three separate payment instruments, requiring 3 signatures for credit hash chains or coin-flip commitments. Even with the multiple-chain PayTree, an entire new chain is needed for each SP and, after a few calls, all new chains may be exhausted. In addition to this enormous flexibility, our scheme has the same payment material storage cost at the user as a chain usable at 1 SP in a single-party prepaid scheme. The pricing contract, which allows dynamic tariffs, adds to SP storage space, although it is still considerably less than some of the hash chain derivatives such as UOBT.

The extra enforcer certificate similarly doubles certificate storage over systems with only user certificates. However, by only issuing SP certificates there are many orders of magnitude fewer certificates in the system,

reducing the communications overhead associated with managing, distributing, and revoking over a billion user certificates, or alternatively shared keys.

The communications cost of multiple independent micropayments is higher than a single multi-party payment. For ordinary prepaid material, including hash chains and Millicent, the broker needs to be online for each new SP, with three times the communications and computational overhead. Use of the enforcer eliminates this, and extensions in Chapter 7 allow the enforcer to be selected offline in our scheme. With credit schemes, the three separate user-generated payment instruments are larger than our single enforcer signed contract, even with the embedded broker signed commitment. In addition, the user traffic, which may pass over a wireless link, is far less than three independent initial payments. An exception to this is SVP, which uses small unsigned messages due to relying on a global shared secret present on a smart card held by every entity.

For an ongoing payment our user traffic is only one hash value, sent in one message, the same as a single-party payment in all the hash based micropayments. As stated in Chapter 4, this is the minimum message size for secure micropayments, and it is kept at 20 bytes despite the number of entities being paid in our protocol. For a call with 3 SPs other schemes would have to send three different hashes from three different chains. Schemes such as SVP, which requires three messages per payment, and schemes where change must be passed back to the user are less suitable due to latencies and co-ordination difficulties in a multi-party scenario. Amongst SPs two hashes must be passed in our scheme, regardless of the number of SPs. For more than 3 SPs this makes the message size less than *N* independent payments.

The computation performed by each party during the first payment for each micropayment scheme was shown to vary considerably in Figure 4-11. The most computationally efficient schemes, such as MicroMint, and Wheeler's bets, are not secure in a multi-party environment. Neither are the efficient shared secret schemes of SVP or Millicent, due to a single global secret and use of high value change respectively. Every other scheme requires over 1000 hashes for a single vendor, and therefore over 3000 hashes for 3 SPs. Our multi-party scheme requires the equivalent of only 238 user hashes regardless of the number of SPs. The SP computation is approximately the same as ordinary micropayments, while our enforcer computation is greater. The extra computation is due to the enforcer signature, which provides the flexibility, security, and other improvements of the protocol.

The user computation for ongoing payments is kept identical to that of a payment to one vendor using a single-party hash scheme. Therefore, with *N* SPs it becomes *N* times more efficient. In addition, the value of prepaid hashes are fixed in the single-party scheme when they are bought from the broker. This often results in multiple hashes being required to pay a certain value, whereas in our scheme the value per hash is dynamically set so that only one hash need be released, keeping hashing to a minimum. Our scheme requires 2 hashes per SP, double that of a basic hash chain, but fewer than schemes such as MicroMint or Millicent. However, two hashes is fully acceptable considering a commodity computer can perform more than half a million a second, as shown in Figure 4-5.

The multi-party micropayment protocol allows micropayment performance to be optimised for paying several entities simultaneously. The user performance costs are kept constant despite the number of payees, and are equal to a single-party payment. With the optimised protocol, the consequences of paying an additional SP is only to add another 12 bytes to the pricing contract, with no additional computation. In contrast with a single-party system, each new payee requires a new payment instrument and an additional performance cost equal to the original one-party payment.

5.11 Summary

We have shown the need for multi-party micropayments in both existing and emerging scenarios. With a large number of network operators and value-added providers it is necessary to guarantee payment and remove the complex trust relationships involved in billing. The desirable properties of a multi-party mobile payment system, which eradicates the problems identified with current billing, were laid down. These requirements are not met by any other payment protocol, and a multi-party micropayment protocol which securely achieves these goals was designed and presented. In addition, the functions of traditional CDRs, other than billing, are not lost.

To allow an efficient solution with minimum computational cost during a call, a scheme using hash chains and off-line broker contact was constructed. Micropayment research has exclusively concentrated on providing payment to a single vendor at any one time. Our solution allows efficient payment to multiple parties with dynamic tariffing and variable charging schemes. The user is presented with exact tariffs before a call, unlike traditional mobile billing where the price may remain unknown. We have eliminated the need for user authentication and online contact with a remote home location. By removing the need for user certificates, the computational load for the user has been reduced, user trust dismissed, and the number of circulating certificates remains manageable. In addition, it removes online certificate status checks with the home issuer and provides a desirable level of anonymity and pricing. Use of an enforcer in the call route allows unspent value to be re-used, even by roaming mobile users moving from one network to another. Unlike traditional billing, the scheme allows fraud prevention to be de-coupled from the clearing process.

A security analysis of the protocol allowed it to be further strengthened, and it was shown that double spending and stealing of value was prevented. This makes it more secure than all payment systems with post-fact detection. This is more sensible for a scheme with billions of users. Performance optimisations were applied, making the protocol even more lightweight and practical. The additional computation required over CDR billing is a small price to pay for the elimination of online communication, inter-entity trust, roaming agreements, and billing fraud. The solution was shown to perform better than using multiple single-party payments, and it requires only a single hash value to be released by the user for ongoing payments.

As mobile communications become increasingly sophisticated and ubiquitous it is critical that the foreseeable problems of CDR billing are addressed. As a first attempt at multi-party micropayments, our protocol seems to be a flexible, secure, and practical answer. We now continue our exploration of the protocol by implementing a prototype demonstrator as described in Chapter 6. A number of extensions and adaptations to increase flexibility and allow use in further application scenarios are proposed in Chapter 7.

6 Multi-Party Micropayments Demonstrator

"A little experience often upsets a lot of theory."
Parkes Cadman

6.1 Overview

In order to validate our multi-party micropayment protocol and to gain experience, a demonstrator was designed and implemented. The majority of micropayment schemes described in the literature, and surveyed in Chapter 3, have not been prototyped. However, the whole purpose of micropayments is to allow scalable repeated payments and only when such a protocol is experimented with in practice can its usefulness and practicality be fully understood. A prototype provides the chance to learn how micropayment theory translates into performance in laboratory trials. The purpose of our prototype was to provide a working demonstrator, showing how multi-party micropayments can be used to pay for ongoing voice and data traffic through multiple independent service providers. Through experimentation, the performance of the multi-party protocol, running on commodity PCs, was quantified and analysed. In addition, the prototype allowed us to assess the suitability of the implementation technologies used for use in micropayment applications.

We chose to implement our prototype using Java [GJS96], due to its portability and wide availability. Java is becoming available on all platforms and devices, both fixed and mobile. Its portability allows applications to be written once, and run on any device with a Java runtime environment. For example, JVMs are already available for Palm PDAs, Symbian communicators and smartphones, and I-Mode mobile phones. The ETSI Mobile Station Application Execution Environment (MExE) standard [ETSI00h] will use Java to run applications on mobile clients. Sun's Mobile Information Device Profile (MIDP) [Sun00] also aims to provide Java on mobile phones. Therefore it is likely that Java will become the standard application environment used in the ubiquitous mobile scenario presented in Chapter 1. The performance of cryptographic algorithms implemented in Java was investigated in Chapter 4. While slower than other more native languages, Figure 4-5 shows that very reasonable speeds can be obtained, especially with hashing.

We used Baltimore's J/Crypto v.3.0 [Bal98a] to supply Java implementations of the SHA hash function, RSA digital signatures and certificate functionality. We had already measured the performance of this implementation in Chapter 4, which allowed us to estimate system performance in Chapter 5 before actual implementation.

The chapter commences by describing the communications architecture and protocol stack, including wireless links, used at each entity. The mechanisms used to pass messages between entities, and with components such as a smart card, are described. Section 6.3 then goes on to discuss further the threads of execution that make up each entity process, and the possible messages that can be passed between threads within a single entity, and between entities themselves across the network. In Section 6.3.1, the payment objects used to hold different payment information, such as hash chains, commitments, and pricing contracts,

within the prototype are examined. The features of the graphical user interface, used to interact with and demonstrate multi-party payments, are explained in Section 6.4. A JavaCard, which is a smart card incorporating a slimmed down Java Virtual Machine, was integrated with the prototype. Its purpose and functionality in the payment system are imparted in Section 6.5. The demonstrator allowed multiple parties to be paid for relaying the user traffic for any popular Internet application. Section 6.6 describes how each SP can control the flow of such traffic, and terminate it when payment ceases.

Finally, a number of experiments were performed to gauge the real performance of the first implementation of multi-party micropayments. The purpose of these experiments was to learn how a micropayment protocol behaves when implemented. The performance and throughput of the basic payment transactions are investigated and, from these, the number of new calls and ongoing payments that can be accepted every second are derived.

6.2 Communications Architecture

The protocol stack used to pass payment messages between entities in our implementation is shown in Figure 6-1. The protocol stack for user application traffic is discussed in Section 6.6. We implemented four different payment modules, one for the broker, one for the SP including enforcers, another for the user, and the final one on a user smart card. For our experiments we used three SPs, of which the first in the call route is the enforcer. The user can physically connect to the local SP using a Bluetooth wireless link, 802.11 wireless LAN protocols, or an Ethernet connection. The implementation is not limited to these datalink protocols as any network card that interfaces with TCP/IP can be used. The SPs and broker, which are all present in the fixed network in our scenario of Chapter 1, connect to each other over a TCP/IP connection. In our testbed environment, this connection is provided over an Ethernet link between hosts.

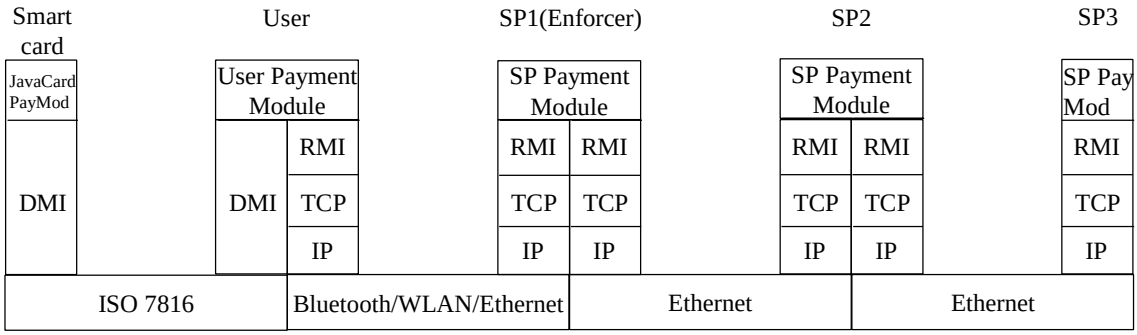


Figure 6-1 Protocol Stack of Payment Control Plane

6.2.1 Inter-entity Communications

Since each application module was implemented in Java, we used the Java Remote Method Invocation (RMI) [Sun98a] as a means to handle message passing between the different entities. RMI allows one to invoke a method of another Java object running on a remote JVM across the network. Basically, it provides flexible remote procedure calls (RPC) based on objects. In our solution each payment module is both an RMI client, in that it calls methods on remote payment entities, and a server, in that it allows local object methods to be called by other remote payment applications. RMI uses object serialisation to marshal and unmarshal data objects into a stream of bytes for network transport. Therefore, RMI provided us with a convenient and rapid way to implement network message passing, without having to worry about the encoding and decoding of those messages across heterogeneous machines. RMI uses TCP/IP [Com00] to provide reliable communications between endpoints.

To make a remote method call, requires obtaining a reference to the remote object from an RMI naming server running on the same machine as the object. Upon starting up each of our payment modules started an RMI registry server, if one was not already running. Each entity registered their identity, such as SP1, SP2 or Mobile1, along with a Listener object that we designed to receive RMI network messages. A sender obtains a reference to the remote entity's Listener object from the remote RMI naming server, and then uses our *sendMessage()* method to pass a Message object across the network, as shown in Figure 6-2. Since the sender is blocked until the remote method returns a value, the Listener thread quickly passes the Message into a shared queue, before terminating the remote call.

The Message is placed into a queue for network messages and the main payment module is notified that a new message has arrived. If the payment thread is not occupied it will process the message immediately; otherwise it will be picked up when it finishes its current task. We constructed a MultiQueue object to represent multiple message queues, allowing messages to be received not only from the network, but also from other local threads such as a GUI, timer, or traffic monitor. A separate buffer is allocated for each source of messages. Synchronised methods in MultiQueue ensure that each method is completed without interruption, and prevents concurrency problems such as race conditions on writing to the queue buffers.

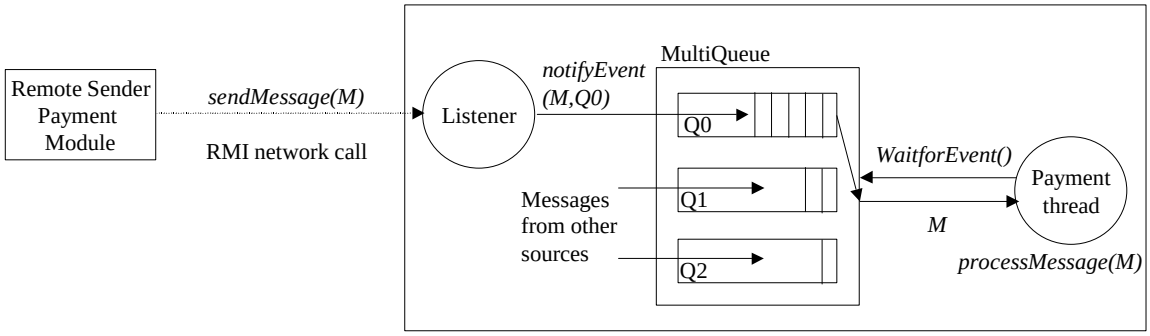


Figure 6-2 Receiving Payment Messages using RMI

Shared queues are also used in the sending of messages, as shown in Figure 6-3. The main payment thread that produces the message puts it into one of several queues, depending on the destination. For each SP there is a separate queue and sender thread for sending to an upstream and a downstream entity in a particular call session. There is also a queue for messages passed to other local threads such as a traffic meter or GUI. These queues allow multiple simultaneous sends, and prevent the payment thread being blocked while waiting for an RMI call to return.

For sending RMI messages we have two different thread classes, Sender and SenderOnce. The Sender class performs a single lookup in a remote RMI naming service to obtain an object reference which it then uses until told otherwise. This is suitable where several messages must be sent to an entity, such as the downstream and upstream entities in a paid call. The SenderOnce class performs an RMI lookup each time it sends a message and does not cache the remote object reference. This is more suited to the situation in which a single message is sent to an entity, such as buying a payment chain or redeeming tokens from a broker.

The Message object is designed to contain any possible message sent within the payment system, including payment messages sent across the network, and inter-thread messages within the same module. The Message object contains the type of message, the sender identity, and another object representing the specific message type. Messages classes were defined for each possible payment message, and are listed in Figure D-1 in Appendix D. For example, the Setup class contains the call setup information and payment commitment from the user. The PayEndorse class carries both payment and endorsement tokens.

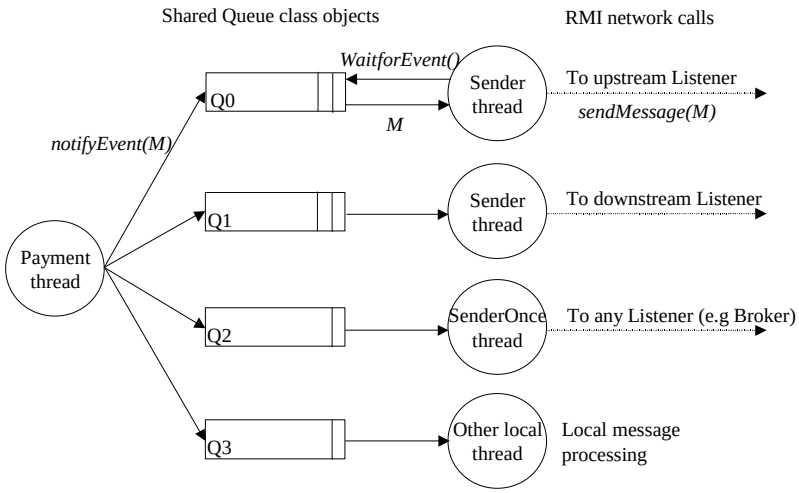


Figure 6-3 Sending Payment Messages using RMI

6.2.2 JavaCard Communications

Our implementation also included a GemXpresso JavaCard [Gem98, VV98], from Gemplus, which was used to securely store and release payment tokens, as discussed in Section 6.5. The JavaCard physical interface with the smart card reader used ISO 7816-3 [ISO97], and the reader was in turn connected to a host PC via a serial cable. ISO 7816-3 defines the communication protocols between a smart card and the physical reader, allowing inter-operable smart cards. The format of basic terminal-card commands and their responses, sent as application protocol data units (APDUs), are defined in ISO 7816-4 [ISO95]. The JavaCard 2.0 specification [Sun98b, Sun97], as used by the GemXpresso card, defines how APDUs interact with JavaCard applets.

The GemXpresso Direct Method Invocation (DMI) protocol was used for communications between the JavaCard and the user payment module. DMI provides an RPC interface to card applets and their methods. Like RMI, it takes care of encoding and decoding APDUs sent between the terminal and card, thereby simplifying development. Each JavaCard applet method requires a specific APDU to invoke it, which DMI abstracts to a single method call. A comparison between using the original JavaCard APDU protocol approach, and the higher-level DMI remote object approach can be found in [VV98].

6.3 Entity Processes and Messages

A system diagram showing all possible messages that can be passed between entity processes, and between threads within a single entity, is shown in Figure 6-4. We do not show the listener and sender threads, or the queue objects, used in sending and receiving these messages. The messages sent between the entity payment processes, labelled R1 to R8, are RMI network messages. The messages U1 to U4 are sent between threads within the user payment module, and the S1 to S3 messages between the components of a single SP. The role and components of each message are given in Appendix D, along with SDL state diagrams for each of the entity payment threads.

The user payment module consists of three other threads, in addition to the sending and receiving threads. The user payment thread handles the main payment processing; a graphical user interface (GUI) thread is used to interact with the user, as described in Section 6.4; a third thread monitors when payment tokens need to be released based on elapsed time or volume of traffic sent. The smart card payment applet is contacted using DMI calls, rather than using our Message class.

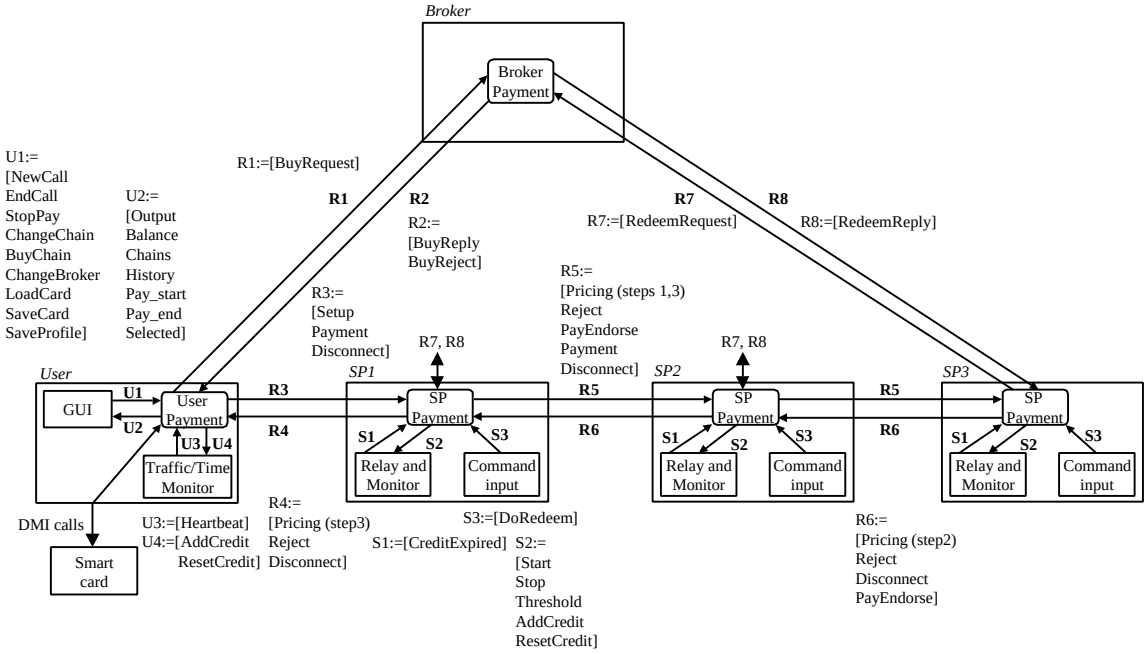


Figure 6-4 Message Paths within the Multi-Party Payment Prototype

The SP payment module similarly uses three more components, other than those used for sending and receiving signalling messages. The SP payment thread contains functionality for both a normal SP and an enforcer, since every SP is potentially an enforcer. The SP must be able to control the flow of user application traffic, based on whether timely payment has been received or not. The relay process can be instructed to allow or deny such traffic to pass. A monitor thread is used to keep track of elapsed time or traffic volume, and informs the SP when the user credit obtained from the last payment has expired. The SP can then close access through the relay. The integration of the relay and monitor with user application traffic is discussed in Section 6.6. A simple command line interface is used to instruct the SP to contact a broker for redeeming collected chains.

When each SP process is started, they are passed parameters that tell them to which other SPs they are directly connected. These are logical connections since for our tests all SPs are physically present on the same LAN. Small static routing tables are used in each SP to calculate which SP to pass a call setup request onto for a given destination. The destination address is included in the pricing contract for this purpose, as well as to create a signed record of the call. As shown in the state diagrams in Appendix D, each SP node passes on pricing and payment messages to the appropriate downstream or upstream node.

Broker-signed X.509 certificates were issued to each SP in our system. During testing it was assumed that each SP already possessed the certificate of other SPs in the call route, from which the public key could be obtained to verify the appropriate signature. However, a variation of the pricing message allows SPs to distribute their certificate as the pricing contract is constructed, if this is not the case.

The broker payment module consists of a process that handles buy requests from users, and redeem requests from SPs, following the protocols established in Chapter 5. The broker must detect double spending and prevent double redeeming of tokens. For each redemption from an SP, the broker verifies all contract signatures; verifies the commitment signature; verifies that the payment and endorsement tokens belong to their respective chains; ensures that the pricing contract has not been redeemed by this SP before; and checks that this claim, together with earlier claims against the same chain, does not cause more than the total value

of the chain to be redeemed. The broker must also handle contracts being redeemed in a different order than they were created. To satisfy these constraints the broker maintains two hash tables:

Table #1: Pricing_contract -> Redeeming SPs
Table #2: Payment_chain -> Total redeemed

Instead of storing each entire contract or chain as the hash table key, a hash function of the contract, and the anchor hash token of the payment chain, are used instead for storage efficiency.

The prototype framework was designed to be flexible rather than produce the optimised performance necessary in a commercial deployment. It allows new messages to be added quickly, or the contents of existing ones to be modified. This proved useful as the payment protocol underwent several revisions.

6.3.1 Payment Objects

A number of objects were designed to represent payment items in the system. Classes were designed to encapsulate a payment token, a cached payment chain, a payment commitment, a pricing contract, and a digital signature. A class may include one or more other payment objects. For example, a payment commitment contains both payment token and signature objects. Classes were also designed to store and manipulate groups of payment objects such as all the payment information after a call, or after a new chain purchase.

By using this object approach, we could redefine the internal structure of different payment structures without affecting other objects. For instance, we were able to easily switch from using a 16-byte MD5 hash payment token to a 20-byte SHA hash payment token, by changing only the PayToken class. A class diagram for the payment objects used in the system is given in Appendix D.

For testing, we used the SHA hash function, which results in 20-byte payment hashes, and 1024-bit RSA signatures. Figure 6-5 shows a textual summary of the contents of, and processing performed on, a PayEndorse message that was sent from the enforcer to the other SPs. It shows the 20-byte payment and endorsement hashes, in hexadecimal form, which have been verified as hashing to the tokens received in the previous message. This is the fifth payment received during this call, as seen from the position of the endorsement token in the endorsement chain. A payment chain is being used from which 5 tokens were already spent before this call began, and hence the reason for the payment token being in the tenth position in the payment chain. The payment results in an additional credit of 5000 milliseconds being extended to the user.

```
Message from: //134.226.38.156/Provider2
PayEndorse message received:
Good paytoken, at position 10
PayToken value: d51f5bb1018b50f4426ddca12eed9152a9292dc2
Good endorsetoken, at position 5
EndorseToken value: 9f6edbfa527da9876cc0f483940e7700db31f9f
Credit is: 5000
```

Figure 6-5 SP Output having Received a PayEndorse Message

The PayChain class was used to keep an array of evenly distributed cached hash chain tokens. Experiments showed that the most efficient configuration was to keep every token in a chain cached in this structure, rather than performing a number of hashes from the nearest cached token. This is due to the large memories

in the laptops and workstations that we were using, where an entire cached chain of length 1000 only required 20,000 bytes and could be kept entirely in primary memory. However, for user devices with less memory, or an enforcer with many simultaneous connections, cache points can be configured appropriately in the PayChain object.

Within the system, we need to be able to load and save payment information. A user needs to be able to store new or partially spent payment chains, and each SP needs to be able to store payment information gathered during a call. The Abstract Syntax Notation One (ASN.1) [ISO90, Lar00] provides a means of describing data structures in a machine independent fashion using a small number of basic primitives. The distinguished encoding rules (DER) are used to encode an ASN.1 structure into a machine independent array of bytes. From our payment object classes we defined ASN.1 structures containing the data attributes of each object and functions to convert a class instance into ASN.1 formatted data. The J/Crypto libraries provided functions for manipulating and DER encoding basic ASN.1 structures. By building on these we were able to save and re-load all necessary payment information in DER encoded format. By saving in a machine-independent format, payment information could be moved between heterogeneous computer systems, such as might occur when importing spent pricing contracts to a traffic analysis or fraud detection application.

6.4 User GUI

One of the main goals of our implementation was to provide a working demonstrator of how multi-party micropayments would function in a mobile networked environment. We therefore wanted to provide a simple and intuitive graphical user interface (GUI) for buying and spending payment tokens.

6.4.1 Balance Counter

Szabo [Sza96] points out that the user interaction with a micropayment scheme should not require more effort than the actual micropayment is worth. For this reason, we provide a simple graphical balance counter that runs alongside any user applications, displaying the micropayment value held by the user. As the user makes voice or data calls, the value of the counter decrements in real-time as payment tokens are released. This allows the user to see how much they are being charged, unlike roaming in current mobile networks. Such a counter also has the advantage that it can be easily displayed on mobile devices with small screens.

Figure 6-6 Screenshot of Micropayment Balance Counter with Voice-call Application

A screenshot showing a mobile user making payments for a voice telephony call is given in Figure 6-6. The voice call is placed using our NOMAD [ODTP98] application, which is based on a popular Internet Telephony package. A NOMAD call is shown in progress, while the counter decrements as payment hashes are released to the multiple parties providing the call. We envision that the average user of our system will not even need to know about payment chains or broker commitments. From their point of view, they top up monetary value in their mobile device or smart card from an online broker or bank, and then spend it as they make calls, monitoring the remaining value through the balance counter. This is similar to the current simple payphone displays that show the amount of credit units or monetary value remaining, decrementing this value as a call proceeds.

The handling of multi-party payments can all be done behind the scenes by the software. User profiles are used to store information about the user's default broker, and calling patterns. Based on these, the user payment application can automatically purchase chains on the user's behalf as needed, and select the appropriate chains for making calls.

Commercial deployment scenarios might allow credit chains to be issued to a trusted user from a home broker, rather than using prepaid value. The balance counter could increment, showing the amount of value spent, rather than the prepaid value remaining. As credit hashes are redeemed by the SPs, the broker can deduct payment from a user account. The amount of credit extended to the user is directly controlled by the number of chains the broker issues to that user.

6.4.2 Full-featured GUI

While a balance counter may be all that is needed by many users, we wanted to provide a flexible GUI for purchasing, viewing, selecting and spending chains in our prototype. Figure 6-7 shows the different components of the full GUI, where the scenario is again a user making a call using NOMAD. The system can be used with any other TCP/UDP based applications, as explained in Section 6.6.3.

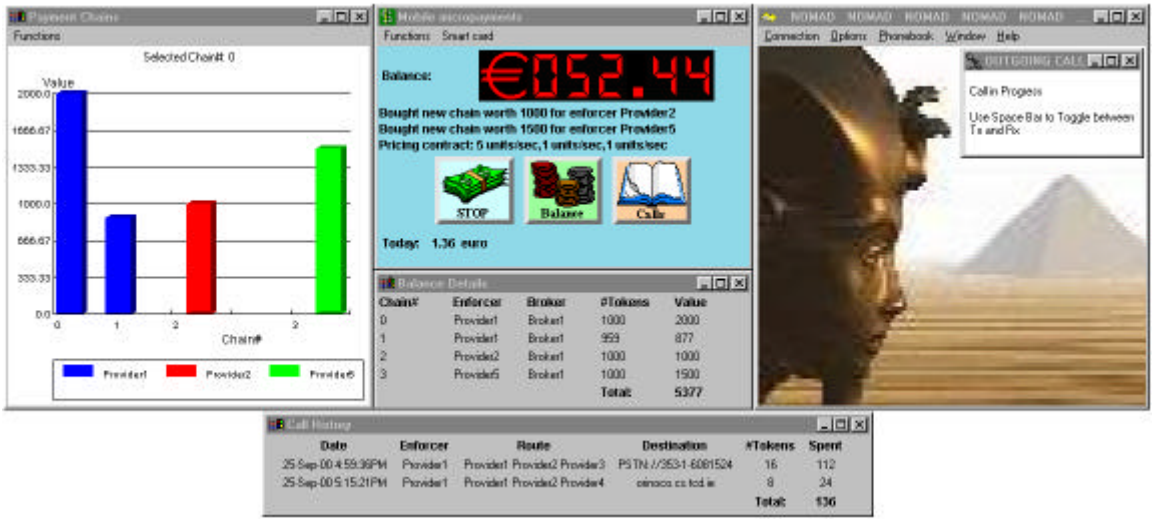


Figure 6-7 Screenshot of Multi-Party Micropayment GUI with NOMAD

The counter in the centre of the screen shows the current balance, in Euro, for the client micropayment application, as before. The user can switch between the full GUI and just the counter alone, as shown in the earlier Figure 6-6, by clicking on the counter digits. Directly below the counter are alert messages from the user payment thread, informing the user about chain purchases and pricing contracts. The Output message is used to send information from the payment thread to the output window, as shown in the state diagram in

Figure D-3. The three buttons below the counter are used to stop payment hashes being released, view the chain balance details and display the records of previous calls respectively. Between calls, the Stop button is replaced with a Pay button. This can be used to agree to a new pricing contract, although by default pricing contracts under a specific price threshold are automatically accepted. The total amount spent by the user on the current day is shown below the buttons, allowing the user to quickly keep track of spending.

A drop down menu in the balance counter window allows further windows to be opened, which accept parameters for making a new call/connection, buying a new chain, and changing the default broker. The same menu is used to open up the chain graph and call history windows. The smart card menu allows tokens and profiles to be loaded and saved to a JavaCard, as described in Section 6.5.

The graph to the left of the screen is a visualisation of the payment chains currently being held by the mobile user. In this case, there are two chains where Provider1 is the enforcer, worth €20.00 and €8.77 respectively. Some tokens have already been spent from the second chain. There is also a chain worth €10.00 which must be spent through Provider2 and another worth €15.00 specific to Provider5. Each chain is identified locally by a short number, and the number of the default chain that will be used, if possible, to pay for the next call is shown above the graph. A drop-down menu on the graph allows the selected chain to be altered. If the enforcer is not present on the call route, the software will attempt to select a chain with an enforcer that can be used.

The payment chain details are also given in text form in the balance details window below the balance counter. The length of each chain is also displayed, and the default is 1000, although this can be set by the user before purchase. The graph and balance details are updated after each call. Clicking on the calls button in the balance counter window brings up the call history window shown at the bottom of the screen. This provides a record of the time, route, enforcer used, and cost of previous calls, providing the user with more information than would appear on a traditional telephone bill.

The GUI thread and payment thread communicate via the U1 and U2 set of messages shown in Figure 6-4. The purpose and parameters of each individual message, and conditions under which they are sent by the payment thread, are given in Appendix D. The simple balance counter seems most appealing for the average user, hiding completely all details of the micropayment scheme. The full GUI provided configurable control, and summary presentation of both chains and calls, allowing efficient management of user payment.

6.5 Smart Card Component

In Chapter 3 we showed that many payment systems use the tamper-resistance of a smart card to provide part of the system security. Examples included stored value smart cards, where a balance is kept on the card; electronic cash schemes, where the smart card prevents user double spending of coins; and electronic cheque schemes, where the card prevents the user overspending their allowed credit. Such schemes also use smart cards to securely protect and transport user secret keys.

While our scheme does not require use of a smart card, we would like it to be able to work alongside macropayment schemes that do. For example, a customer might use a Common Electronic Purse Specification (CEPS) [EV99] card to make a macropayment to the broker to purchase multi-party chains, which can then be stored and used from the same *multi-application* card. In addition, by storing payment chains on a smart card, we provide convenient portability. A user can carry multi-party value on a smart card in her pocket, and use it from any device, mobile or fixed, that is equipped with a smart card interface. For example, the same card could be used to pay for calls from a mobile phone while travelling on foot, used to

pay for services from a car computer [Hei99], used to pay for items through a public kiosk, and used to pay for information from a fixed device at home or in the office.

With this in mind, we used an optional multi-application JavaCard in our prototype. Since JavaCard is multi-application, we envision a macropayment applet being implemented on the card alongside our multi-party scheme. For example, VisaCash has been implemented as a JavaCard applet [BBEH+99]. We used a prototype GemXpresso JavaCard from Gemplus with a GCR410 universal smart card reader. The GemXpresso card uses a 32-bit microprocessor, based on an ARM7 design, and provides 512 bytes of RAM, 32 kilobytes of EEPROM and 8 kilobytes of ROM. The GemXpresso Rapid Applet Development (RAD) environment allowed JavaCard applets to be converted to handle the GemXpresso DMI protocol, discussed in Section 6.2.2, and downloaded onto the card.

A subset of JavaCard 2.1 [Sun99] functionality is implemented on the prototype card. Unfortunately, the cryptography functions are not fully implemented and no cryptographic co-processors are present. A current high-end deployment card with full functionality can produce the cryptographic speeds given in Chapter 4. Simple substitute test algorithms are provided on the prototype instead of DES, 3DES, and SHA. For example, the SHA hash function is replaced with a simple left shift by one byte. No implementation is provided at all for the asymmetric RSA algorithm present in the JavaCard 2.1 specification.

We implemented secure storage of payment chains and a user profile on the smart card. We required a user PIN to be passed to the card in order to access it, with multiple incorrect entries preventing further access. When the user payment GUI was started, it probed the local host to see if a smart card reader was attached. If present, a smart card menu was made available through the GUI, which allowed the user to access and control a JavaCard after authenticating to it with a PIN. Payment chains and hashes were saved to the card in the same ASN.1 format used when saving to local disk, as described in Section 6.3.1. Since the GemXpresso DMI protocol is limited to handling messages of 64 bytes or less, we wrote routines to fragment longer messages and re-assemble them on the card itself.

A user profile could also be saved onto the card, and this included call history and user preferences such as default broker, preferred enforcer, and spending limits. When a user arrived at a new terminal they could simply insert their JavaCard and their preferred settings and payment value is loaded into the new device.

We also implemented payment chain hashing on the card using the dummy SHA function. Our initial idea was to only release parts of a single payment chain to an untrusted terminal rather than the full value. However, this required a method for the card to differentiate between trusted and untrusted terminals, which ultimately required terminal authentication based on signatures. Since the GemXpresso card does not have asymmetric cryptography, we did not explore this option further.

Instead, we used the card hashing to implement a one-time signature used for offline enforcer selection, as described in the Section 7.2. The user passed the card a number representing the enforcer, and the card returned two hashes that made up the one-time signature. The two short signature chains are created by the card, on request of the user at chain purchase. We also used the dummy SHA hash function to create the CardMAC necessary for card authentication. Since the card does not use real SHA hashing, we had to use the same dummy SHA hash function to verify the signature and MAC in both the user and SP payment modules.

While no real cryptography was present on the prototype card, it did show the feasibility of using a JavaCard to provide secure storage and portability for our scheme. There was a noticeable transfer delay when communicating with the card, which is quantified later in our experiments. This latency would make it

unsuitable for directly releasing tokens from the card with a frequency of less than a second. Having assessed the suitability and capability of a smart card component, we describe the design of several enhancements that depend on it in Chapter 7.

6.6 User Traffic Control and Application Inter-working

In exchange for payment, each SP will transport user traffic through its network and pass it onto the next SP in the call route. Figure 6-8 shows such a scenario where the user traffic passes through one or more routers or switches within each SP domain. Depending on the charging mechanism, each SP needs to be able to *monitor* certain characteristics of the call, such as elapsed time, traffic volume sent and received, and QoS provided. For each payment token received, the user is extended a certain amount of credit, as specified in the pricing contract. For example with a voice call, this might be another 10 seconds of conversation, or with a data connection it might be another 10KB of IP traffic with priority level 2.

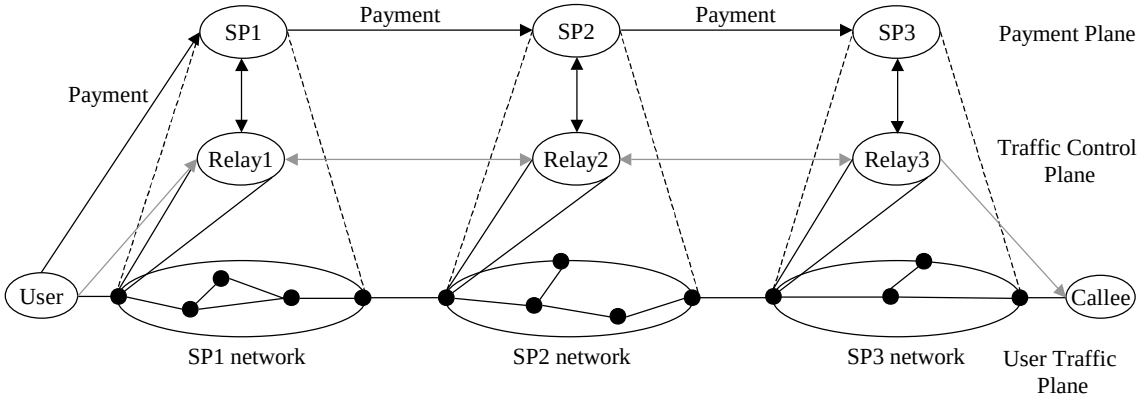


Figure 6-8 Payment Integration with User Traffic Control

When the credit expires and no further payment is received, or a bad payment token is obtained, the SP needs to terminate the flow of user traffic through its domain. How this is done will depend on the network architecture, but it might involve disconnecting a call at the switch, no longer relaying traffic, or degrading the QoS from a high priority level down to a best-effort service. A suitable place to put the SP monitor and relay control functionality is where the call enters the SP's domain, as shown in Figure 6-8.

Logically, the call flows through each of the relays. For our prototype, we implemented and associated a call monitor and traffic relay with each SP in the call route, and forced user traffic to pass directly from one relay to the next. The SP payment module informs its monitor how much credit to extend to a flow and, when this expires, the relay terminates the call. The design and integration of the relays with the payment modules and user application traffic is now examined.

6.6.1 Voice Traffic Relay

Currently, the most popular use of mobile devices is for voice calls. To demonstrate our payment system with a voice call, we implemented a traffic relay for our NOMAD IP telephony application [ODTP98]. NOMAD uses the User Datagram Protocol (UDP) over IP for both signalling and voice traffic. UDP provides unreliable transport, but does not have the setup overhead or re-transmission delays associated with TCP. It is used largely in streaming audio and video applications, where minimising end-to-end delay and jitter are more important than the safe arrival of every packet.

To avoid having to change the NOMAD application code, we wrote a UDP relay application in Java which forwarded all traffic on the UDP ports used by NOMAD. The traffic can be forwarded in turn to another

relay or onto the final callee, as shown in Figure 6-9. The relay process is controlled by a payment process using the S1 and S2 sets of messages of Figure 6-4.

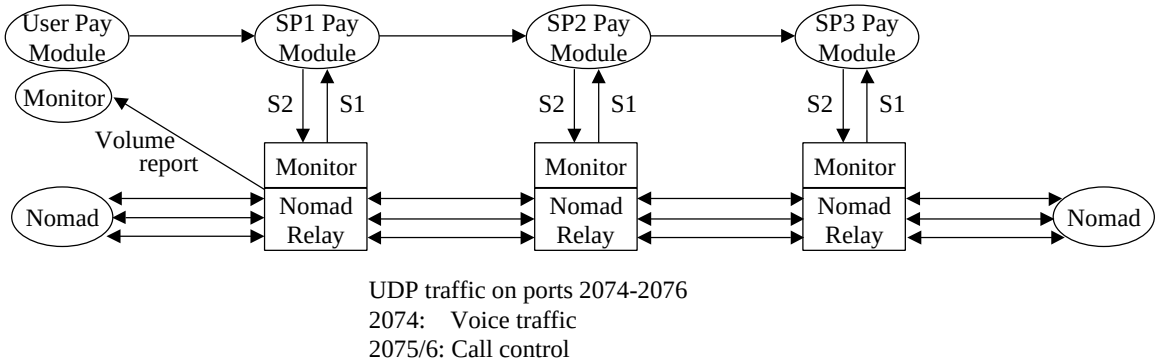


Figure 6-9 Integration of Voice Traffic Relays

The disadvantage of using this simple approach is that the NOMAD user, when setting up a new call, must enter the address of the NOMAD relay rather than the actual destination, so that the UDP traffic is directed to the relay. In turn, the relay must be instructed as to where to forward the traffic. For a more transparent integration, proxy relay configuration could be integrated with NOMAD.

Current implementations of the IP stack introduce noticeable delays for IP telephony calls. This problem is compounded by passing the traffic through the IP stack at each relay. To minimise the delay for demonstration purposes, we used a single relay entity, controllable by every SP in the call route. Any SP can stop the relay, but it required all the SPs to start it. This provides the same logic functionality as each SP controlling its own relay, but without the extra delays. The relay was initially informed of how many SPs were controlling it using the Threshold message of Figure 6-4.

6.6.2 Monitors and Heartbeats

A monitor thread at each SP keeps track of the call, based on time or volume. The current call credit held by the monitor thread is incremented by the payment module each time a valid token is received. For time-based charging, the monitor thread is simply a timer, and it is credited with a number of seconds. For volume-based charging, the monitor thread decrements a volume counter as packets are received and forwarded. In both cases, when the credit has run out, the payment module is informed and forwarding of traffic through the relay is halted.

The user payment module also needs to be informed when to release payment tokens. It monitors conditions locally and releases tokens according to the pricing contract. A monitor thread, similar to those used by the SPs, is employed to monitor time or traffic volume, and send a *heartbeat message* to the payment module when the next token needs to be released. This functionality is shown in the state diagram for the user process in Appendix D. For time-based charging, a timer is used. For volume-based charging, a local relay can be placed on the user host, through which user traffic passes, which informs the monitor of the volume sent and received. Alternatively, for efficiency, the SP relay can also inform the user monitor of traffic volumes.

6.6.3 Generic Traffic Relay

We envision multi-party micropayments being used to pay not only for voice calls but for any application that generates network traffic, voice or data based. For this reason, we wanted to validate our prototype with any TCP/UDP application available. Each application protocol usually uses a specific set of well-known

ports for communication. One option is to write a TCP/UDP relay that forwards traffic on these well known ports, as we did with NOMAD. A Web proxy also works on this principle forwarding Web HTTP requests, usually on port 80. However, in order to interwork with any existing and future applications we need to eliminate application-specific knowledge at the relay. The method we chose to do this involved using a SOCKS server.

SOCKS [LGLK+96] is a networking proxy protocol, designed to transparently and securely allow application traffic to traverse a firewall, without need to configure the firewall for every application type. It encapsulates application messages in a SOCKS message which is then sent from the user host machine to a SOCKS proxy server at the firewall. There it is decapsulated and a connection made to the destination host across the outside network. The SOCKS proxy relays both TCP and UDP traffic between the source and destination hosts. Most popular applications can be configured to use a SOCKS proxy. For those that cannot, programs exist which can dynamically intercept network calls from a normal application and redirect them to a SOCKS proxy, a process known as *socksifying.*

Since the Socks protocol provides a method of relaying any application traffic, we decided to use it in our solution. Rather than integrate our relay control and monitor into an existing Socks server, we designed our own relay between the user application and the Socks server. Our relay forwarded all Socks traffic in both directions, and was controlled by the SP as before, as shown in Figure 6-10. A volume monitor was able to tally the application traffic size. As before, when payment stops, the SP halts the relay and the application traffic, carried in Socks messages, cannot pass.

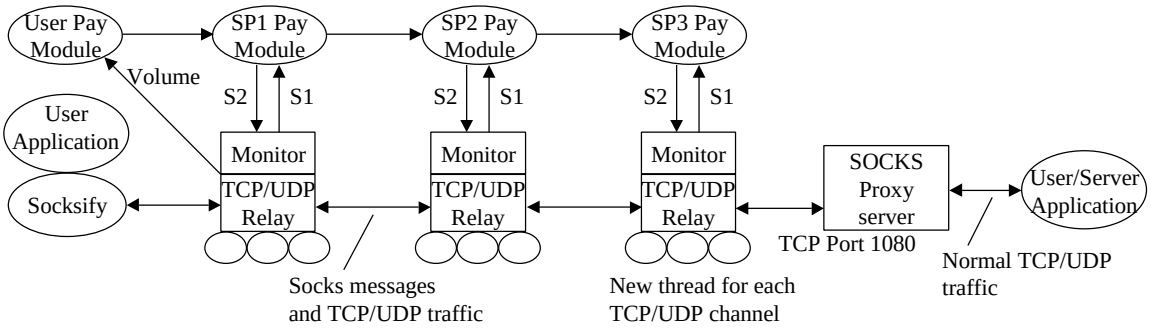


Figure 6-10 Integration of Generic Traffic Relays

We used SocksCAP from NEC to socksify applications with no Socks support built in, and configured it to communicate with our first relay. We also used their Socks server to handle the Socks traffic from our final SP relay. Again, where speed was critical, a number of independent relays could be logically replaced with a single relay controlled by all the SPs. Socks v.5 supports relaying of both TCP and UDP traffic, with a TCP control channel used to setup and maintain a Socks UDP relay on the Socks server. By closing the TCP control channel passing through our relay, the UDP relay on the Socks server was also closed. When using time-based charging, this allowed us to stop UDP traffic without having to setup our own UDP relay based on the port numbers agreed on in the control channel.

The Socks server can also relay UDP traffic initiated by the destination, such as occurs with many streaming video and audio applications. For this to work, the client application must set up a TCP connection to that destination, and the socksify program informs the socks server of the port that the client expects traffic to arrive on. Unfortunately, NOMAD uses only UDP traffic, and therefore could not be used in this scenario, but required its own special relays designed in the previous section. In a commercial implementation, the relay of traffic would be controlled at the network layer in a router, rather than at the transport layer. Since

Java does not currently allow direct access or manipulation of IP network packets, we did not explore this option further in our prototype.

We were able to demonstrate multi-party micropayments from a wireless laptop with many popular Internet applications including RealAudio (streaming video and audio), Internet Phone (IP to PSTN telephony) Netscape Navigator and Internet Explorer (any data type over HTTP), Yahoo Messenger (instant messaging and voice telephony), and mIRC (text chat). We were able to do sample charging based on time or volume, with traffic termination once payment ceased. These demonstrations showed the feasibility of using such payments and demonstrated how they can be practically presented to the user through a balance counter, without distraction or interruption from the main application.

6.7 Experiments and Measurements

In order to investigate the performance of our multi-party payment prototype we took a number of measurements at each different entity in a running system. The total time to perform specific actions, such as buying a chain, establishing a contract, or spending tokens, was measured. The percentage of the total time spent performing computation at each entity, and communicating between entities, was quantified. Our measurements are based on the original multi-party micropayment protocol, before additional optimisations were applied, as described in Chapter 5.

6.7.1 The Experimental Setup

We used four host machines in our testbed, one 400MHz Pentium II machine with 196MB memory running Windows NT, and three 233MHz Pentium laptops with 32MB memory running Windows 98. We chose Bluetooth and WaveLAN as two typical wireless technologies that we envision providing high speed cheap access for roaming mobile users, as described in Chapter 1. By employing real wireless links with our prototype, we were able to gauge its performance in the envisioned mobile environment.

We used one of the laptops to represent a mobile user, and used Bluetooth, WaveLAN, and Ethernet network cards with it. Another laptop was used to run SP1, the local SP for the mobile user. Since Bluetooth is peer-to-peer, a Bluetooth network card, in addition to Ethernet, was also used in SP1. A second WaveLAN card on the laptops was not necessary because a central WaveLAN basestation received and forwarded traffic onto the local network, where it could be received over Ethernet. We used Digianswer Bluetooth democards [Dig00] to provide the Bluetooth technology [Blu99, Haa98]. A Lucent Technologies WaveLAN card provided IEEE 802.11 [IEEE99] wireless LAN technology.

A broker was also run on the same laptop as SP1, in order to be reachable directly over Bluetooth. We used the fixed 400Mhz host to run SP2, and the third laptop to run SP3. A typical call in our scenario originated from the mobile and passed through SP1, SP2, and SP3. Calls were established and paid for over Bluetooth, WaveLAN, and Ethernet, as shown in the protocol stack of Figure 6-1. We ran SP2 on the fastest machine, because it represents a core network operator which would typically have more processing power than an operator at the edge.

We used the Java 1.1 Microsoft JVM, as it was required by the Gemplus JavaCard, and J/Crypto required Java1.1. From our cryptographic benchmarks in Chapter 4, this meant that our RSA signature verification and creation would be faster, but our SHA1 hashing slower than other JVMs. Thus it would make starting a new call faster, but verifying payments during a call slower. Use of the Microsoft JVM means that our computational results cannot be compared directly with those estimated using the Sun JVM benchmarks in Chapter 5.

We used the *Java System.currentTimeMillis()* for millisecond timing. While timestamps on the 400MHz host were of milisecond granularity, the laptops only provided time with an accuracy to the nearest 10ms. Hence all measurements with a short runtime were timed for 1000 executions and, from this, the time for one calculated. We took the same precautions as when performing benchmarks in Chapter 4, by forcing garbage collection and turning off J/Crypto obfustication before testing.

In Chapter 3, we explained Yen's assumption of (n-1)/2 hashes being required per hash chain payment, due to only keeping the chain root in memory. We then applied this in Chapter 4 when analysing micropayment schemes. During experimentation, we used chains of length 1000, and we found it more efficient to hash the entire chain and keep each hash value in memory rather than re-compute from the root for each payment. This was because our host machines were equipped with plenty of memory and a full chain only required 20KB. However, on a smart card or small mobile phone at the user side, or an enforcer with tens of thousands of simultaneous calls, full caching may still not be possible.

6.7.2 Wireless Network Round-Trip Times

Our measurements involve sending payment messages over different wireless links. We therefore wanted to know how much delay is introduced at the network layer, to understand the minimum extra time that will be added to every message sent, depending on the network transmission used.

We used the Packet InterNet Groper (PING) utility to send out a single Internet Control Message Protocol (ICMP) echo request packet from the mobile to SP1 and vice-versa. The average round-trip times for sending the network packet over Bluetooth, WaveLAN, and Ethernet are shown in Table 6-1.

Network	Round-trip time (ms)
Ethernet	1
WaveLan	14
Bluetooth (Slave to Master)	22
Bluetooth (Master to Slave)	32

Table 6-1 Round-Trip Network PING times for Bluetooth, WaveLan and Ethernet

An interesting observation is that there is a significant difference in round-trip latency depending on which Bluetooth entity was the source. Transmissions initiated from slave to master took on average 10ms, or 31%, less time than traffic sent in the opposite direction, from master to slave. We made the mobile the slave, and SP1 the master, as would be the case in a real-world scenario. In our system, most of the payment messages flow from the mobile slave to the master SP.

Fixed-line Ethernet added only a 1ms latency, while WaveLAN added 14ms, and Bluetooth added up to 32ms. However, WaveLAN timings were much more varied than Bluetooth, with an occasional latency as high as 67ms. This extra delay may have been due to sharing bandwidth to the WaveLAN basestation with other users. These figures only give us an idea of the minimum latency delay that will be added to every payment message. Our system uses TCP and RMI layers over the network layer, both of which add additional headers and send acknowledgements, perhaps resulting in more than one round-trip for each payment message sent. Packet size is increased not only by these layer headers, but also with the inclusion of our payment payload. When a payment message is sent, we would expect the additional network delay to be a multiple of the ping times, depending on the number and size of actual packets sent.

6.7.3 Computation

As we showed in Chapter 3 and Chapter 4, one of the key design goals of a micropayment system is to keep the computational cost to minimum, especially for repeated payments. In this section, we discuss the results

of measuring computational components within our system. We investigated the time taken to generate chains, to buy from a broker, to sign and verify a contract, to create an endorsement, to make and check ongoing payments, and to redeem spent chains.

6.7.3.1 Chain generation

A user generates hash chains before sending the chain anchor to the broker for signing. An enforcer generates a shorter endorsement chain per call. Both entities can generate chains offline in bulk before they are required. Table 6-2 shows the time taken to generate hash chains of different lengths on both the 233Mhz and 400MHz machines. It took 57ms and 39ms to generate a chain of length 1000 on the mobile and SP2 respectively. For very short chains, the majority of the time is spent creating the payment chain object rather than performing the small number of hashes, which is why the time to generate a chain of length 100 is only approximately twice that to generate one of length 10.

Chain Length	Time (ms)	
	Mobile(233Mhz)	SP (400MHz)
10	4	2
50	6	3
100	9	5
500	31	20
1000	57	39
5000	273	188
10000	541	374
50000	2691	1854

Table 6-2 Hash Chain Generation Times

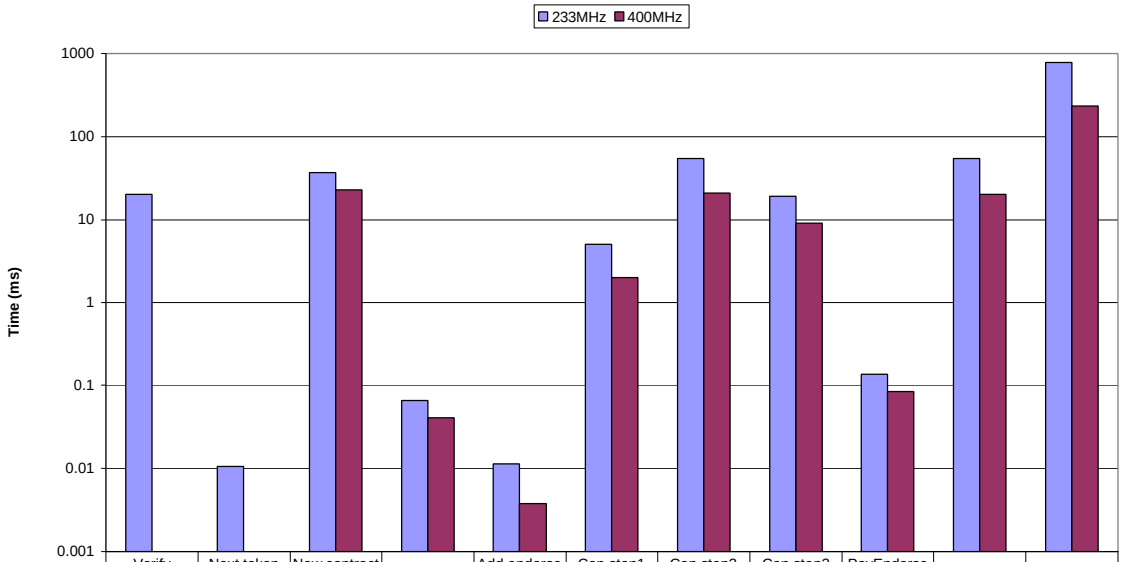
The figures obtained for the 400MHz machine are in line with the performance of J/Crypto SHA1 on the Microsoft JVM as shown in Figure 4-3. With our implementation, a chain of length 10,000 took 374ms, which equates to 26,738 hashes/s, compared to the pure benchmark speed of 33,407 hashes/s. The additional overhead of our implementation is due to creating and allocating space for the payment chain object, and using a random number generator to obtain a chain root. We have shown hash chain generation to be highly efficient, even on the Microsoft JVM, which had the slowest SHA1 speed.

6.7.3.2 Multi-Party Payment Computation

Figure 6-11 compares the time to perform specific computations, such as verifying a fully signed contract, or checking a payment token. Certain procedures are only performed at some entities, and the entity performing the procedure is identified by the label U for user, E for enforcer, or SP for normal service provider. For SP services we show the speed on both 233Mhz and 400Mhz machines, because we had SPs running on both architectures. A logarithmic scale is used to show the large differences.

We start by examining the computational overhead at call setup. The new contract computation by the enforcer SP1 takes 37ms. This time is spent verifying the broker-signed commitment including checking against overspending, constructing a new unsigned contract with a new endorsement anchor, starting a new sender thread to communicate with the mobile, and caching all of the endorsement chain in memory for fast access. We used an endorsement chain of length 500, or half the length of the payment chain.

The bulk of the time was spent hashing out the endorsement chain for caching purposes. Since the whole chain is rarely ever used, this cost could be reduced by only caching the first 100 chain values, which would take 9ms instead of 31ms, based on Table 6-2.



	Verify contract (U)	Next token (U)	New contract (E)	Payment (E)	Add endorse (E)	Con.step1 (SP)	Con.step2 (SP)	Con.step3 (SP)	PayEndorse (SP)	BuyReq (B)	Redeem (B)
233MHz	20	0.01049	37	0.066	0.01137	5	55	19	0.137	55	786
400MHz			23	0.0407	0.00375	2	21	9	0.0844	20	234

Figure 6-11 Time Taken to Perform Basic Computational Multi-Party Payment Procedures

The time for an SP to process each step of assembling the pricing contract is shown. In step1, the broker commitment signature is verified and SP fields added. The operation takes 2ms at SP2, which is in line with the 508 signature verifications possible a second from Figure 4-3. In step2 each SP checks that the contract still contains at least the same values as in step1 and then signs it. It is this signature that makes it the most expensive of all steps and it takes 55ms on the laptop. Finally, in step3, the SP checks that its own fields remain present and verifies all 3 SP signatures, including its own. This takes 19ms on the laptop, most of which is spent checking the signatures. The user also verifies all 3 signatures on the contract, which takes approximately the same time as step3 at an SP with the same computational power.

Step2 is the computational bottleneck, with only 48 step2 computations/s possible on the 400Mhz machine. However, one of our optimisations in Chapter 5 was to remove a normal SP signature from the contract, which would reduce this overhead for everyone but the enforcer. SPs acting as enforcers would be expected to have more computational resources than normal SPs.

The time spent performing computation for ongoing payments is important, as our design attempted to minimise this. From the chart we see the time spent obtaining the next payment token at the user, processing that payment at the enforcer, adding an endorsement token, and processing both payment and endorsement tokens at an ordinary SP, are very small compared to other operations. The user takes a mere 0.01049ms to obtain the next payment token, and this is so fast because it is basically an array lookup to get the token from the chain cache and a construction of a payment message object. It is the same procedure at the enforcer for adding an endorsement token, and takes slightly longer due to the extra token in the new payendorse message object.

The time taken to verify a user payment at the enforcer, and an endorsed payment at an SP are 0.066ms and 0.137ms respectively. These translate to 15,150 payments/s and 7,300 endorsed payments/s on a laptop. In addition to the hash function, a 20-byte array comparison and copy are performed, yielding slower speeds than pure benchmark hashing. These values are far less than the time taken to generate a short new chain because they do not involve creation of new chain objects.

From the timings, we calculate the number of operations/s possible for each computational procedure. Based on the protocol for setting up a new call, an enforcer will compute a new contract, step2 and step3, while the other SPs will perform step1, step2 and step3. Assuming each operation is performed serially, we can estimate the number of newcalls/s possible at an enforcer and ordinary SP. As shown in Table D.2 in Appendix D, we find the enforcer to be the bottleneck with 9 and 19 newcalls/s possible on the 233MHz laptop and 400MHz host respectively. Non-enforcer SPs can handle 13 and 31 newcalls/s on the same hosts.

The computation performed by the broker to process a user buy request and an SP redemption request were also examined. The buyrequest involved generating a broker signature and a response message, and therefore took a similar amount of time to step2 of contract assembly. The redeemrequest involved verifying tokens from ten separate calls, with ten contracts and parts of two different payment chains. While this was the longest of all operations, it can be performed offline when the broker's processors are otherwise idle.

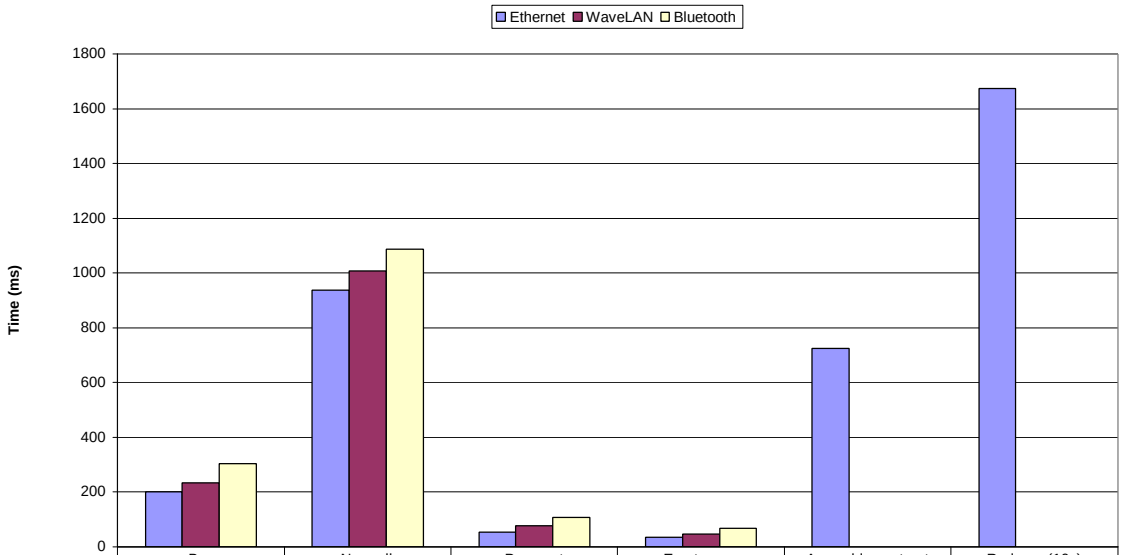
Our untuned Java implementation has yielded computational speeds of 19 newcalls/s and 11,850 ongoing endorsed payments/s on a commodity 400MHz PC. This may be quite adequate for a small home-based network operator, offering local Bluetooth or WaveLAN wireless services to passers-by, as envisioned in Chapter 1. Larger network operators will require more computational resources, which can be provided by parallel processors as might be found in a network switch. As we have quantified the computational cost of the prototype, we now examine the impact of the communications overhead.

6.7.4 Communication

The total round-trip time to perform the standard actions within the payment system were investigated. These included the time it took to buy a chain, set up a new call, and make an ongoing payment. This time included not only the payment computation performed at each entity, but also the communications overhead of constructing a message, queuing it, passing it to RMI to deliver across the network, and placing it in the arrival queue at the receiver. Timings for messages involving the mobile user were made over Bluetooth, WaveLAN, and Ethernet.

Figure 6-12 shows the time taken to perform six basic actions within the system. These actions were buying a chain, initiating a new call, sending a payment, sending an empty message, assembling a contract amongst the SPs, and redeeming spent chains. Some of the actions involved receiving a response, such as the buy request and newcall request, while others, such as payment, travelled in one direction only. Requests made over Bluetooth took longer than those made using WaveLAN, which in turn took longer than those using Ethernet. This is as expected from our initial network round-trip times in Section 6.7.2. However, the additional latency is several times greater than measurements taken at the network layer, and this is due to larger messages and the additional acknowledgements and headers of the TCP transport layer and RMI messaging layer.

The larger the payment system messages passed over the wireless link, the greater the additional latency added by Bluetooth or WaveLAN, over Ethernet. The Bluetooth latency ranged from 32ms, for sending and receiving back an empty payment message, up to 151ms for sending a commitment in a newcall request and receiving back a signed contract. In comparison, the corresponding WaveLAN latencies only ranged from 11ms to 32ms. These delays are not significant enough to affect perceived user performance adversely. As Figure 6-12 shows only a small proportion of the communications delay is due to wireless network processing and transmission.



	Buy	Newcall	Payment	Empty msg	Assemble contract	Redeem (10c)
Ethernet	201	937	54	35	724	1673
WaveLAN	233	1007	76	46		
Bluetooth	305	1088	108	67		

Figure 6-12 Total Time Spent Performing Actions within the Multi-Party Payment System

The time to buy a chain includes constructing a request at the user, sending it over RMI, broker checking and signing of the commitment, and receiving the response back at the user. Timing for a new call setup begins when the user constructs the call request and finishes when the user has verified all three SP signatures on the signed pricing contract returned by the enforcer. The contract assembly time is the time between receiving a user newcall message at the enforcer, and the enforcer adding the final signature to the contract. It therefore includes steps1 and 2 of contract assembly at each SP, and step3 at the enforcer. Contract assembly forms 77% of the total time taken to establish a new call.

The payment time is the time taken for a payment message to pass from the user through all the SPs, with each SP verifying tokens before passing them on. In order to gain an accurate timing measurement of this, both the user and SP3 were run on the same laptop, and therefore shared the same system clock. Tokens can be forwarded before verification but, since they can be verified so quickly in comparison to communications time, we did not pass on invalid tokens.

The time taken to transmit an empty message from the user through all SPs and back to the user again was measured. The message was received, queued, and processed at each SP in the normal fashion, taking only 35, 46 and 67ms for the round-trip over Ethernet, WaveLAN and Bluetooth respectively. While this suggests that the longer time spent by other actions is due to the SP computation performed, this does not explain why the payment action, which requires minimal computation, takes 19ms longer. It is largely due to the additional RMI computation of marshalling and unmarshalling payment tokens, which is discussed further in Section 6.7.5.

The redeem action claimed value for 10 spent chains from a broker running on a laptop. As expected, it is the slowest action, due to the number of chains involved, but would normally be processed offline in large batches.

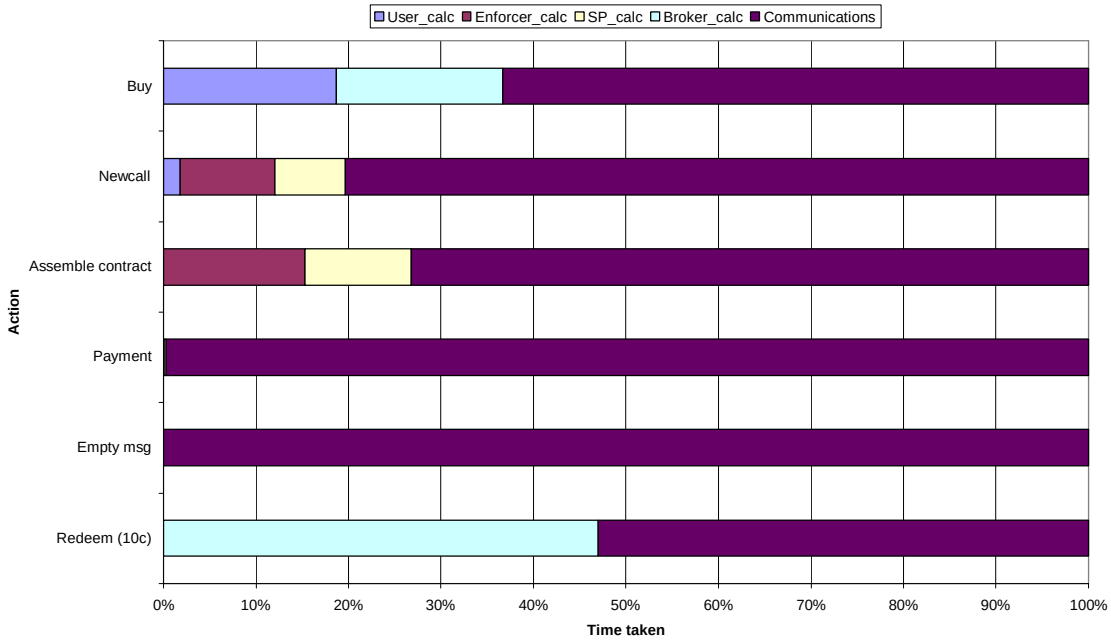


Figure 6-13 Contribution of Entity Computation and Communications to Total Action Time

Each action consists of time spent processing that payment message at each appropriate entity, and a communications overhead. The communications time consists not only of sending the message across the network, but preparing it for sending, queuing in the listener and sender threads, and the RMI marshalling overhead. The stacked bar chart of Figure 6-13 shows how the total time spent performing an action is divided between computation at each entity and the communications overhead. The chart is derived from the computational timings of Figure 6-11, and the user action timings with Bluetooth and non-user action timings with Ethernet from Figure 6-12. This represents our typical call scenario of a Bluetooth user making calls through SPs in the fixed network. However, the stacked chart changes very little for the case where WaveLAN or all Ethernet timings are used, showing the relatively small impact of the wireless technology present. The timing data for each component is given in Appendix D.

The time spent performing and processing communications dominates every action within the payment system, with computation always consuming less than 50% of a transaction time. Due to the low network latencies measured earlier, it can be surmised that Java RMI is responsible for the majority of the communications overhead. Contract assembly processing uses less than 30% of the transaction time, and the total processing for a new call forms less than 20% of the elapsed time, due to the additional Bluetooth latency. The important observation is that the payment message shows no computational overhead compared to the communications time. Nearly all the time is spent sending the message, and less than 0.003% of the time is spent processing it, even though each entity verifies or manipulates it. In addition, the payment communications time was already the second lowest value in Figure 6-12, after the empty message. The communications overhead is the bottleneck for ongoing payments, with several orders of magnitude more payments possible than the RMI communication subsystem can deliver. In the next section, we examine how much of a bottleneck the communications overhead is by attempting to set up and maintain a large number of ongoing payment calls.

6.7.5 Throughput of Simultaneous Requests

While the last section showed the time to perform payment system actions, we did not measure the maximum throughput of the system. So, while it takes a user 937ms to establish a new call, we do not know how many simultaneous calls can be established per second with the prototype. We have figures for computational

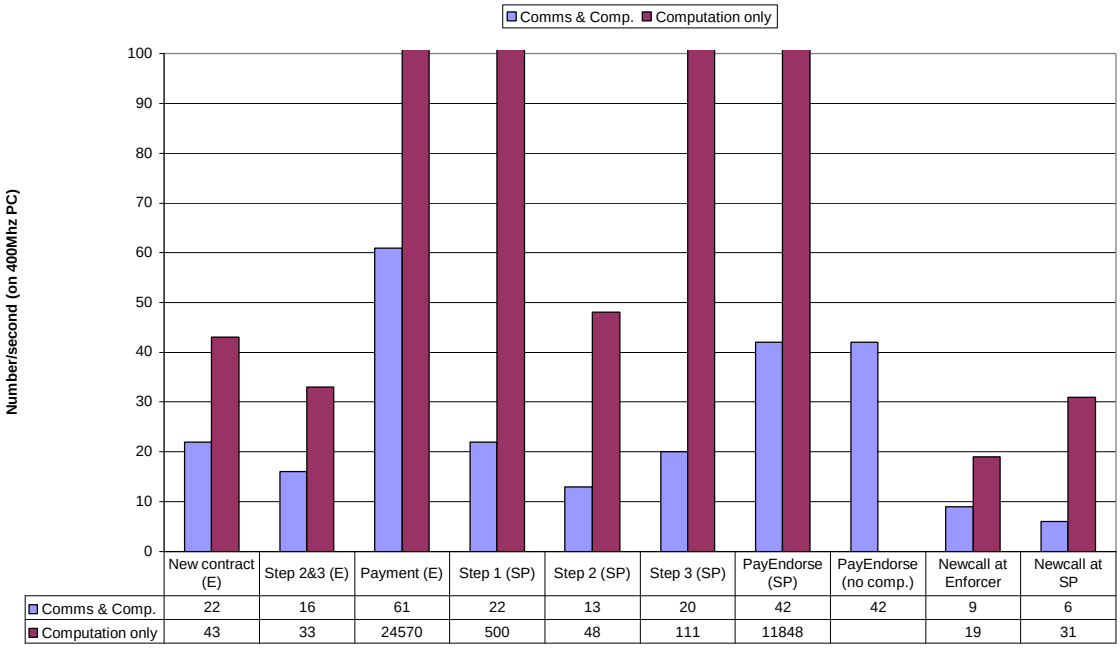
throughput, discussed in Section 6.7.3, but there is a large communications overhead, as shown in the previous section. We now investigate how many calls can be set up per second, or how many payment messages can sent through and processed by an SP per second. While we can control the computational requirements of the system, in that it will depend on the micropayment protocol design decisions, we cannot so readily control the communications throughput. Since there is a communications overhead, the combined throughput measured here will be less than the computational throughput. We want to quantify exactly how much less. Not only will this allow us to benchmark the system, but it will allow us to assess the suitability of using RMI for micropayments.

In order to measure the maximum throughput, we needed to flood the SP with the appropriate payment message, allow it to process the message, and pass it onto the downstream entity. We ran an SP entity on the 400Mhz PC and bombarded it with RMI messages from four different laptops, with four sender processes on each. When the receiving SP is at its maximum throughput, the CPU usage will be 100%, at which time it will not have any additional computational cycles to dedicate to additional message processing. The message requiring the least computation to process was the payment message holding a single payment token from the user, as shown in Section 6.7.3. It required the full output of four laptops with four senders each to obtain 100% CPU usage on the SP for this message, while other messages did not require so many senders before computational saturation. We also needed to ensure that the CPU usage on each sending laptop was kept below 100% saturation so that it did not become a sending bottleneck, with the SP waiting for it to send more messages. Instead, our setup allowed each laptop to have the capability to send as many messages as the SP could possibly handle. Thus, the next RMI message was immediately sent as soon as the previous RMI sending call returned.

We also needed to monitor the size of the incoming network traffic to the SP, in order to ensure that the network did not become saturated and hence become the bottleneck. The incoming network traffic rate did not go above 383 KB/s, and the average outgoing traffic rate from each laptop was 86.4 KB/s. The 10MB Ethernet never went above 35% network usage during the tests. Since the network was not saturated, the throughput was limited by the processing at the SP. We used Netstat Live from Analog X, and NetXRay from Network General, to obtain real-time network statistics during testing.

While CPU computation is time-sliced between the multiple threads that make up the SP payment module, as detailed in Section 6.2, we needed to ensure that a huge number of pending messages did not build up in the receiving or sending queues. We limited each queue to hold 1000 messages and, once the queue became full, the appropriate thread that was filling the queue yielded until it was at least half emptied. When the receiving queue was full at the SP, the senders were blocked, as the RMI method call was not returned. This effect was clearly noticeable by examining the output traffic rate at each sender.

In Section 6.7.3, we measured the time to perform certain payment procedures. A specific message triggers each procedure and another message is sent in response. For instance, a call setup request from the user, causes the new contract procedure to execute at the enforcer, who in turn sends a pricing contract step1 message to the next SP. To test the throughput, we sent the trigger message using RMI to the SP, who performed the appropriate computation, before passing it over RMI to the next SP. We measured the interval between every 10,000 messages leaving the SP. After starting to send messages to the SP, we allowed several 10,000 message intervals to pass before timing began, in order to allow queues to fill and to measure typical behaviour rather than performance immediately after initialisation. By measuring the time over a large number of intervals we were able to obtain an accurate measurement of the average time taken per action, and from this calculate the throughput as the number of procedures/s at the SP.



	New contract (E)	Step 2&3 (E)	Payment (E)	Step 1 (SP)	Step 2 (SP)	Step 3 (SP)	PayEndorse (SP)	PayEndorse (no comp.)	Newcall at Enforcer	Newcall at SP
Comms & Comp.	22	16	61	22	13	20	42	42	9	6
Computation only	43	33	24570	500	48	111	11848		19	31

Figure 6-14 Prototype Throughput for Payment Procedures with and without Communications

Figure 6-14 shows the throughput obtained for payment procedures when both communications and computations were used. The chart also compares these results to the original throughput with no communications overhead present. Steps 2 and 3 of contract assembly are performed together at the enforcer, if first in the call route, and is faster than receiving and processing both steps on their own. To set up a new call, the enforcer will receive and perform both the new contract, and later the step2&3 procedures. In turn, the other SPs must receive and process step1, 2, and 3 of contract assembly, each with a communications overhead. By combining these separate measurements, the number of newcalls that can be processed can be found. With our RMI prototype, the enforcer can handle 9 new calls/s, but an ordinary SP can only handle 6 calls/s, on the 400 MHz machine. Even though the computational overhead of the enforcer is greater, the other SPs must receive an additional message, the overhead of which slows down their throughput.

For ongoing calls, the enforcer can handle 61 payments/s while SPs can handle 42 endorsed payments/s. The difference in these figures is not because payment processing of endorsed tokens takes longer, but because the RMI messages of endorsed payments are longer and therefore take more time to process in the RMI communication subsystem. We see this is the case from the chart because, when a PayEndorse network message is sent but no payment processing is performed on it at all at the SP, the throughput is still exactly the same, with a value of 42. Payment processing is so efficient that it has no impact on the throughput of our prototype.

The actual throughput for each action is considerably less than the computational throughput. However, by far the greatest reduction in throughput is experienced by the ongoing payment messages. The communications overhead reduces the maximum possible number of payment messages by three orders of magnitude. Payment requests were processed serially by each SP in our prototype. Where user response times are critical for call setup, a number threads could execute in parallel to process requests. This will allow the system to handle bursts of new calls gracefully although it will not improve throughput.

We also ran throughput experiments with an SP running on the slower 233MHz machine. The incoming network traffic was considerably less at 125KB/s. This is due to the slower processing of RMI requests, and hence fewer RMI network calls/s.

The ICEBERG [WRCB+00] project at the University of California at Berkeley is designing and implementing an Internet-core network architecture for integrated communications. While they do not consider real-time payment, they do use Java RMI for call setup within their implementation. On a high-end 500MHz PC their prototype handles 10 call initiation/s over RMI, which is similar to our prototype performance when adjusted for the 400MHz PC.

We have shown that the additional time experienced over computational times is not due to network saturation. Java RMI keeps TCP connections alive between hosts, and therefore it is not the transport layer setup times that introduce the delay, but rather the RMI message processing. Java RMI is slow due to processing object type information at runtime as part of marshalling, or serialisation, of objects. In addition, 38 new objects are created for every remote method invocation. On the SP we observed three new threads being created for each incoming call from a new host. This is due to RMI supporting parallel method invocation on the server side for clients on different hosts. Therefore, our Listener thread was replicated, although all messages were still passed into the same single queue for payment processing. Ideally, to improve user response times, a pool of payment processing threads would collect messages from the queue. However, RMI latency can be halved by improving object serialisation [NPH99]. Speed improvements of several orders of magnitude can be obtained by using native compilation, and by pushing the run-time overhead of RMI to compile time [MNVB+99].

While RMI provided a useful means to pass payment messages across the network in our prototype, in its current form it heavily limits the throughput of the protocol, especially for ongoing payments. Alternatives are to use improved versions of RMI or to assemble network packets directly. In contrast, Java cryptography implementations were shown to perform fast enough for micropayments.

6.7.6 JavaCard Measurements

The performance figures in Chapter 4 show a high-end JavaCard with maths co-processors to have cryptographic capabilities only one order of magnitude slower than a commodity PC. However, there is no indication of how other operations such as storage or communications with the card perform. Since a prototype JavaCard is used within our system to store payment chains and perform limited hashing, we wanted to quantify the performance of these operations. The time to load and save the user profile and a varying number of chains to the card was investigated.

Item on JavaCard	Bytes
System files	3232
Gemplus Library	472
Payment applet	745
Free code space	6496
Chain storage	4400
User profile storage	64

Table 6-3 Usage of Storage Space on the JavaCard

Table 6-3 shows a breakdown of how memory space on the card was utilised. The card application space is limited to 11 KB of bytecode. Our applet, which performed PIN security, loading and saving of chains and profiles, and one-time signature hashing, took up only 745 bytes after optimisation. However, it did rely on some Gemplus utility libraries which took up 472 bytes. There was plenty of remaining space, 6496 bytes, for further applets. We used up all the user storage space on the card, allocating 4400 bytes for chain storage and 64 bytes for a user profile. User preferences can be efficiently encoded in a profile, and hence the initial small size. We were able to store 20 DER encoded payment chains onto the card, which used up 4364 bytes.

Only the highest token in a chain is stored along with the commitment, keeping each chain a fixed size despite its length.

User storage space not used by an applet is used by the processor stack, along with 512 bytes of permanent space. Since we used the majority of user storage it was thought that the reduced stack size might impact negatively on the card's performance. However, experiments with reduced chain storage space, and hence increased stack size, did not affect experimental results with our applet.

The time to load and save a varying number of chains to and from the card are shown in Figure 6-15. The timings include making a connection to the card, presenting the user PIN, performing the action, and then disconnecting from the card. They do not include the time spent encoding chains or writing to disk on the host machine. Chains were deleted from the card after they had been successfully loaded. Due to time spent connecting to the card it was faster to load the user profile when loading the chains, than to load the two separately.

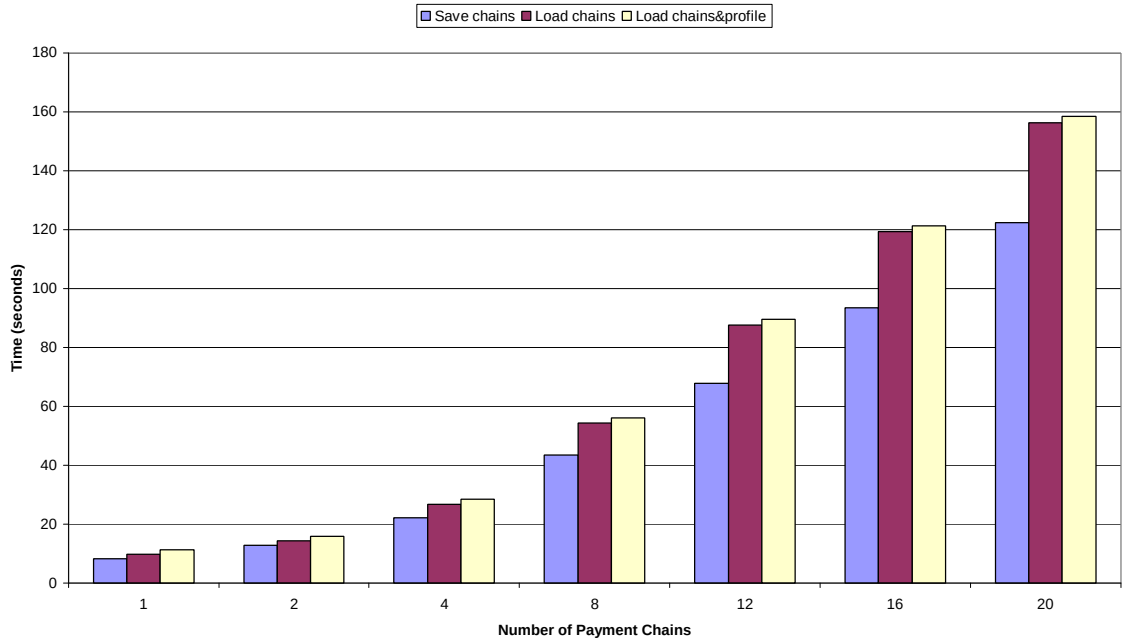


Figure 6-15 Time to Load/Save Payment Chains on JavaCard

Writing to the card was consistently faster than reading from it, as shown in the chart. This was due to having to allocate a new temporary byte array on the card to copy the chains into when loading, and also, to a lesser extent, zeroing the contents of storage after a successful load. We envision a typical user carrying a small number of payment chains, perhaps between two and eight. It took 16 and 28 seconds to load the user profile along with 2 and 4 chains respectively from the card. Since this operation only needs to be performed once at the start of the session, it is an acceptable overhead. However, it took 94 and 119 seconds to save and load 16 chains respectively, which is unacceptably long for many users, especially if chains are bought through a physical ATM machine or at a Point-of-Sale (POS) terminal. The time to save and load a user profile alone was found to be a much more reasonable 3.9 and 4.8 seconds respectively, due to its small size. The JavaCard timing data is summarised in Appendix D.

There is an overhead of 2 seconds in connecting and disconnecting from the card. Each of our load/save operations makes a new connection to the card, performs the operation and then disconnects. This delay can be eliminated by keeping the connection to the card open throughout a user session.

The Gemplus DMI protocol, discussed in Section 6.5, limits the size of data in the card APDUs to 64 bytes. Thus, to save a chunk of data to the card requires splitting it up into 64 byte blocks and making a number of repeated serial calls to the card to save each block in turn. A single chain is DER encoded in 221 bytes, and therefore required 4 calls, one after the other, to save it. The greater the number of chains, the more calls that need to be made, either for loading or saving, and hence the long communication times measured.

It was found that a newly initialised card, onto which applets had just been downloaded, performed better than one that had been in use. The performance diminishes by up to 20% over the first ten uses after initialisation. This is most likely due to the temporary stack space becoming full and having to move objects off the stack as further methods are run. Our measurements are taken after the card had been in use and therefore reflect more realistic timings.

Although a prototype JavaCard was used, the slowness of writing more than 8 chains to the card is likely to be a prohibitive factor for use in a commercial setting, no matter what cryptographic speeds can be attained within the card itself. Applications that write more than 2,000 bytes to the card will need improved speed, which may be attainable by manipulating APDUs directly rather than through the DMI protocol.

6.8 Summary

We have implemented the multi-party payment protocol and successfully demonstrated how multi-party micropayments can be used to pay for a wide variety of typical user applications. It was shown how each SP can be paid in real-time for transporting user traffic and how that flow can be monitored and terminated once the user discontinues payment. The prototype was tested in the envisioned mobile environment with wireless links, showing that users can roam into a mobile network and pay for services without need to contact distant home networks. A message passing architecture with queuing was used to exchange payment between network entities and between threads within these entities.

Payment for voice calls, the current dominant mobile application, was demonstrated. This was augmented by demonstrating ongoing payments for a variety of other popular Internet applications, which we envision being used by roaming mobile users. The payment scheme was also executed over fixed wireline connections, showing its suitability for deployment in both fixed and mobile scenarios. A simple GUI was used to hide the payment details from the average user, and still provide the flexibility required by advanced users. A JavaCard was used to carry payment chain value, which could be loaded onto whatever communications device was currently in use. Our working solution allows dynamic pricing of services at each provider, removes the need for online authentication of a roaming user, replaces the trust relationships of traditional billing with guaranteed payment and non-repudiable proof of a call for all parties involved.

Our prototype measurements show that the system is efficient enough to handle frequent small multi-party payments. The bulk of the computation takes place in generating the contract signatures at call setup, although the system bottleneck was the RMI communications overhead, over which we had no control. We expect that the communications processing can be reduced by several orders of magnitude in a commercial implementation where payment packets are passed directly to the network layer.

Telephone usage statistics show an average arrival rate of 2.8 calls/hour and an average call duration of 2.6 minutes/call, during peak usage [LCW97]. Based on this model, it would require fewer than 50 commodity 400MHz PCs to handle micropayment computations for the call traffic in a city the size of Dublin, with a population of one million. This assumes a mean payment rate of 10 seconds, and uses our computational rates of 19 newcalls/s and 11,848 endorsed payments/s. Having shown the multi-party micropayment

protocol to be viable, we develop it further in the next chapter by providing extensions which make it more flexible and allow it to be applied in a wider range of scenarios.

7 Ubiquitous Multi-Party Payments

"Man's mind stretched to a new idea never goes back to its original dimensions."
Oliver Wendell Holmes

7.1 Introduction

A first multi-party micropayment protocol designed for mobile communications was presented in Chapter 5. It addresses the problems of current mobile billing systems making the scheme a suitable replacement for billing in both existing and emerging mobile network architectures. The solution allows all the independent untrusted parties providing voice or data services to be paid in real-time.

While Chapter 6 showed that the protocol works well in this basic scenario, there are several other situations and environments in which it could be applied. It would be desirable to be able to use the same payment chain tokens to pay for local services, such as using the nearest printer, buying items from a vending machine, or paying at a point-of-sale terminal. In addition to paying service providers, users will want to pass monetary value to their friends, colleagues, or even complete strangers. It should be possible to use payment chain value to pay any party in any location at any time, providing truly ubiquitous payments.

In a world where every user will carry at least one mobile communications device, it is desirable to have connectivity to the fixed network constantly available. As envisioned in Chapter 1, dense populated areas will have many independent picocells built from low-cost wireless technologies, such as Bluetooth, which will supplement the second and third generation wide-area mobile networks. Pedestrians are likely to wander frequently from one picocell to another, perhaps during an active call, and the payment protocol needs to deal gracefully with these handovers. Another possible scenario is for the user mobile devices to form temporary connections between themselves, forming an ad-hoc network. There will need to be a payment incentive in order to relay a stranger's traffic to the nearest basestation.

Further considerations involve the fixed part of the network. The multi-party protocol will need to work alongside and interact with networks that use legacy CDR billing. Current network research in providing quality-of-service across central core networks has aimed at reducing the per-flow state maintained. Similar criteria will need to be introduced into the payment protocol to improve scalability in a network that relays hundreds of thousands of ongoing calls.

This chapter proposes and evaluates the extensions and modifications necessary to use multi-party payments in these different scenarios. In some cases the protocol is transformed completely, resulting in an accompanying protocol tailored for a specific setting. Section 7.2 presents a method to allow the enforcer to be decided after a payment commitment has been purchased. In Section 7.3, the protocol is adapted to allow payments for local services such as vending machines. The flexibility is increased further in Section 7.4

where a technique allowing user-to-user payments, such as from one mobile device to another, is described. Mobile ad-hoc network payments are introduced and explained in Section 7.5.

The original protocol is synchronous in that every SP is paid simultaneously. In Section 7.6, we propose asynchronous multi-party micropayments as a means to allow some SPs to be paid while others are not, and explain where this is useful. Following this, an alternative protocol where each party pays only its downstream entity is described in Section 7.7. This indirect payment technique is used in aggregating payments in central core networks to eliminate per-flow state, as explained in Section 7.8. Finally, the last innovation outlines how payments can continue uninterrupted while the mobile moves from one independent picocell to another.

7.2 Offline Selection of Enforcer

A disadvantage of the multi-party micropayment scheme is that one of the SPs involved in a connection, the enforcer, must be known in advance before the payment chain is purchased. This inconvenience is minimised by seamlessly performing a chain purchase at call setup if necessary. In addition, the requirement to use a specific SP is no different to pre-paid phone cards, calling cards or discount calling services.

However, ideally, we would like to be able to nominate the enforcer of a new chain offline when making a new call. This would remove the purchase delay and allow *universal chains* to be purchased, spendable through any SP. The problem with allowing the user access to a payment instrument spendable anywhere is that it can be double spent at an unlimited number of SPs with post-fact detection, an unacceptable risk in a global scenario.

Another option, not present in other payment systems, would be to limit the maximum possible fraud by limiting the number of possible entities at which the tokens could be spent. With a chain of total value v, we could limit the enforcer to be one of c possible SPs, thereby limiting the maximum fraud value to $(c-1)v$, instead of an infinite amount. Such an ability to fix the enforcer dynamically offline, from a set of possible enforcers, might only be given to trusted users who have a lasting relationship with a broker. In order to detect the double spenders, the chain can be linked to the user by including an identity field, blinded or otherwise, in the commitment.

Due to the scalability and administrative problems of blacklisting cheating users, we prefer to place no trust at all in them. We propose that a secure tamper-resistant smart card be used to dynamically fix the enforcer in a payment chain commitment. By limiting the enforcer to one of c possibilities, the maximum fraud is limited even if the card is compromised. We now show how one-time signatures are used to decide the enforcer, and we then discuss how a broker can differentiate between a request from a smart card and one from a user with no card present.

7.2.1 Selection using One-Time Signatures

One-time signatures as a method of using only hash functions to generate and verify a single signature, were presented in Chapter 3. We used the technique to design a hash chain that could be infinitely extended in Section 3.3.3.6. Here, we use a small one-time signature to make a payment chain specific to one of a small possible set of enforcers listed within a chain commitment. We decide to limit the size of the enforcer set to eight, although this number can be tailored to the maximum amount of fraud that the broker is prepared to risk.

When a payment chain is purchased, the user nominates the eight possible enforcers, and their identities are listed within the broker-signed payment chain commitment. The position of each enforcer in that list can be

represented with 3 bits, ranging in binary from 000 to 111. The position of the enforcer, in binary, is used to represent that entity in the one-time signature. Three bits can be represented with a hash chain of length (2^3-1) = 7. We label this chain, used to represent the enforcer number, SA as shown in Figure 7-1.

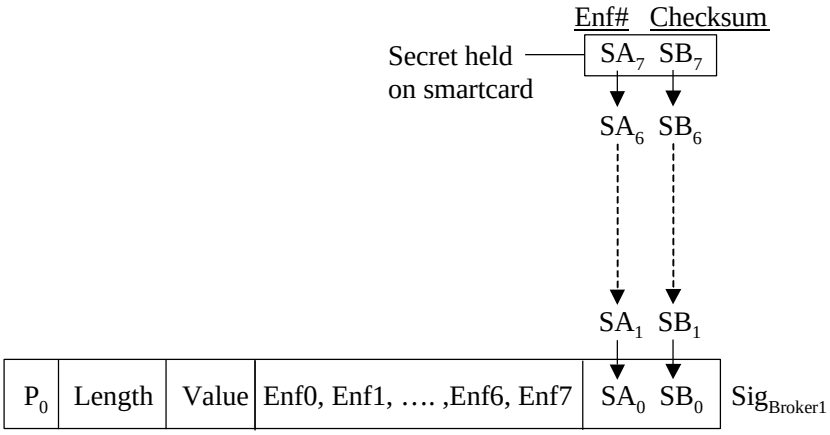


Figure 7-1 Payment Chain Commitment with One-Time Signature to Select Enforcer

To open the one-time signature to sign the chain for enforcer#2, the smart card will release SA_2. However, as explained in Section 3.3.3.6, a checksum is required to prevent the receiver falsely claiming that he only received SA_1 and that the enforcer is therefore enforcer#1. The checksum is represented with another chain of length 7, called SB. The checksum is a count of unreleased parts of the SA chain, and it is this inverse relation that protects the signature. For example, to sign for enforcer#2, the smart card will release SA_2 and SB_5. SA_2 represents enforcer#2 listed in the commitment, and SB_5 states that 5 hashes remain unreleased from the SA chain.

During purchase, the anchors of the signature chains SA_0 and SB_0 are sent to the broker, along with the list of 8 desired enforcers and the original purchase request fields. The chain roots SA_7 and SB_7 which can generate the one-time signature are never released from the smart card. After the signed commitment, shown in Figure 7-1 has been received and the user wants to make a call they will select one of the 8 enforcers. The smart card is informed of their choice, and it generates the appropriate one-time signature. This is passed to the appropriate enforcer in the call request who can verify it, along with the other SPs.

7.2.2 Authentication of the Smart Card

During payment chain purchase, the broker must be certain that the request originates from a smart card and not just a user alone. Otherwise the user could formulate the request, knowing the secret signature values, and later double spend at all 8 enforcers using a different one-time signature for each. The smart card removes user control of the one-time signature.

A practical solution is to allow smart cards with a prepaid chain with a specific possible set of enforcers to be bought offline in a physical shop. For example, one might buy a card with a chain for use with any one of two local mobile networks, two local fixed networks, two popular VASPs and two foreign networks. A card might include several different chains, each with eight different possible enforcers. This removes the online purchase and the need for authentication of the card, and is more flexible than current prepaid phone cards, which are specific to a single SP. However it will not allow the card to be re-used.

The other alternatives, observed in smart card payment schemes discussed in Chapter 3, are to use a shared secret between the card and the broker, or to use a private asymmetric signing key within the card. Secret key authentication, discussed in Chapter 3, can be performed using encryption or a MAC, and a nonce used for

freshness. Whatever the case, all cards should not share the same keys. Instead, individual card keys or, more practically, small groups of shared keys should be employed. In this case, if one card is broken, it can be identified at the broker by linking the authenticated purchase with the doubly spent chains.

We now describe the withdrawal and spending protocols for offline enforcer selection. The smart card is within a small group of smart cards, perhaps one hundred, that share the same key with a specific broker. This reduces the total number of secret keys that must be maintained by the broker and also maintains privacy for the user.

During withdrawal, the broker needs to authenticate the one-time signature anchors as originating from the smart card. A MAC, such as HMAC, will provide this using the shared key. However, it will not prevent a replay of SA_0, SB_0 and their subsequent inclusion in later commitments. This is not a problem though as the smart card will only ever sign the one-time signature once, after which the SA/SB chain roots are deleted. The released signature values SA_X, SB_Y are kept in case of communications failure so that the actual signature is not lost. Even if SA_0 and SB_0 are replayed into a new commitment, the old one-time signature will have to be used, limiting the chain to one enforcer, and preventing double spending at multiple enforcers.

On first inspection, it seems no extra security risk by allowing replays, as the same single enforcer# will be used. However, given a large amount of time, huge computational resources, and some luck, it might be possible for an attacker to find the SA_7 and SB_7 chain roots, or even parts of the chain allowing more than one enforcer to be selected. If the roots are one day discovered and there is nothing stopping a replay of the authenticated anchors into the commitment, this would allow many different commitments each to be spent at up to eight enforcers, by using the same SA_0 and SB_0. Consequently, if replay is avoided but the chains are broken or compromised, the single commitment can only be spent at the limited eight enforcers. This was our initial security requirement.

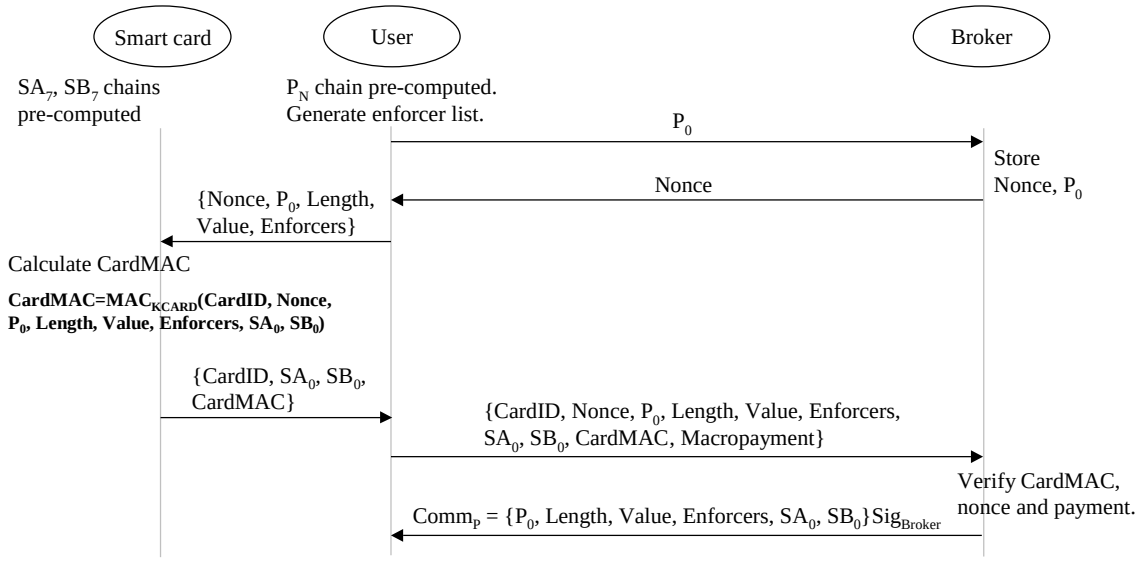


Figure 7-2 Smart Card Interaction for Chain Purchase

Therefore, we must prevent replay by including a nonce from the broker. The nonce cannot be generated by the user, since they are not trusted. The smart card could generate the nonce but then the broker would have to remember every nonce used per card. The number of nonces stored could be limited by including the date and time as part of the nonce but, in general, smart cards do not have an onboard independently powered clock, which makes this infeasible. For these reasons, the broker generates a fresh nonce for a withdrawal request which must be matched up with the smart card reply as shown. Storing a copy of the nonce for the

few seconds duration of a withdrawal request is a scalable solution, even with thousands of simultaneous withdrawals.

The card MAC includes the nonce, the one-time signature anchors, and the card group identity which allows the broker to lookup the corresponding secret key. We also include the payment details in the MAC to link it to the user request. Although these contents are not needed by the card, they provide integrity of the entire request, removing the need for public key encryption used in the original withdrawal. If necessary, the macropayment details may still be encrypted.

The broker verifies the MAC and nonce and issues the new commitment with enforcer list and selector signature. When later needed, the user will instruct the card to sign for the selected enforcer, and this will only ever be done once. The signature roots SA_0, SB_0 are used to identify the particular signature, allowing multiple different commitments and signatures to be simultaneously managed.

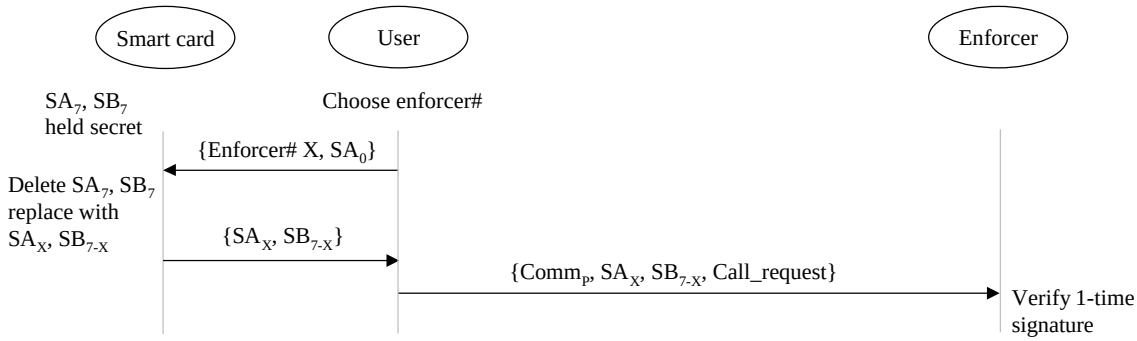


Figure 7-3 Offline Enforcer Selection

Offline enforcer selection has been provided using a small one-time signature and a trusted smart card. The user can decide which of eight possible enforcers to use any time after chain purchase. The computational cost of the selection is 7 hashes on the smart card, and 7 hashes by anyone who wishes to verify the signature. This is several orders of magnitude more efficient than performing an asymmetric digital signature on a smart card, as shown in Chapter 4. An additional 7 enforcer identities and 2 hash values are added to the payment commitment, and, with efficient enforcer identity encoding, this requires less than 60 bytes. If the smart card is compromised, or the one-time signature root discovered, the maximum fraud is seven times the chain value. By linking the card identity during withdrawal to commitments with multiple enforcers, further purchase requests from the card can be denied. In contrast, as discussed in Chapter 3, a break of existing smart card payment systems allows almost unlimited fraud. In summary, we have removed the need for any online broker contact by securely allowing prepaid value to be made vendor specific offline.

7.3 Local Area Payments

Chapter 5 showed how to pay the multiple parties involved in providing a service efficiently. A typical service might be a mobile call spanning several independent networks. However, there are situations in which it may be necessary only to pay one or more entities in the immediate proximity. Examples involving repeated payments include paying to print pages on a nearby printer, or using a local photocopier. A parking meter could be paid every five minutes by an onboard device in a car, preventing the risk of an unpaid parking fine or unnecessary returns to pay for more time. Alternative examples include vending machine payments for everyday items such as drinks, food, and tickets. Similarly, such purchases might be made at a point-of-sale (POS) device in a shop or at entrance turnstyles. In each case, such payments might be made over a local radio link such as Bluetooth or wireless LAN, or may use infra-red or a fixed connection.

While our multi-party micropayment scheme will work with any networked entities, the problem in a local area scenario is deciding who or where the enforcer will be. The simplest solution is to nominate the local entity to be paid, such as the printer, vending machine, or POS, to be the enforcer. With offline enforcer selection, as described in Section 7.2, this can be done spontaneously offline at the time of purchase. The chain will then be specific to that device and a new chain will be needed for each new device encountered. This is acceptable if one regularly uses the same set of vending machines and POS terminals, as might occur during a normal work day. However, it is not satisfactory when one roams into new areas making payments to a variety of new devices that may not be encountered again.

One solution is to place an enforcer in the network between the user and the VASP device. Payments pass out into the local network to the enforcer and back to the nearby VASP. At a high level, this is the model used by existing mobile commerce macropayment protocols described in Chapter 3. However, in those schemes, payment is limited to a single NO who accepts payment on behalf of the VASP. In our local payment scenario, we envision the enforcer covering a local campus, shopping centre, or geographic region. In this way, a prepaid chain for that enforcer could be spent at any of the local VASPs in the area, with a new pricing contract for each different VASP. Here, the enforcer is effectively acting as a local online trusted third party, which may or may not take a commission on effecting the purchases. The enforcer can be set to be any local NO, making it more flexible than existing m-commerce solutions. However, the solution does assume that every VASP is connected to the network, and does not take advantage of a wireless peer-to-peer connection with the user.

7.3.1 The Smart Card is the Enforcer

One of the key ideas of our payment scheme was to avoid involving unnecessary parties, and we would like to be able to keep that principal for local payments. In the previous section, we used a user smart card to choose one of eight enforcers offline. We now propose that the *enforcer entity reside on the smart card*. The enforcer becomes a program on the smart card that cannot be tampered with. The *smart card enforcer* assumes the same trust and characteristics of a real enforcer; it has a private key and matching certificate, signs pricing contracts, and keeps the balance of unspent user chains. Since the enforcer is on the local user held card, it can always be present in any call route. Therefore a single payment chain can now be spent at *multiple SPs*, which provides the offline flexibility needed for local payments.

The card enforcer works on behalf of the brokers for the user. Since, in this case, the enforcer is not a real SP it will not charge for its services, and will place a zero charge field in its section of the pricing contract. Figure 7-4 shows how payments are made to a local VASP using a smart card enforcer. For example, a user might be paying to print files which she is transmitting from her local handheld device to a nearby printer over a Bluetooth link. The user purchases a payment chain as before, with the enforcer field set to be the identity of the smart card enforcer. The user holds the payment chain and sends the call request to the smart card. A pricing contract is assembled containing the charging rate of the VASP, and no charge for the enforcer. For example, the printer might charge one cent per page.

The contract signing messages pass through the user but she plays no part in them, as before. If the user is happy with the finalised contract, she starts releasing payment hashes in order to obtain service. Each payment hash is sent to the card enforcer who verifies it and releases an endorsement hash. The user then forwards both payment and endorsement hashes to the VASP who verifies them and provides the service. The protocol remains unchanged. The user now has the same opportunities for fraud as an outside man-in-the-middle attack, which was shown to achieve no value in Chapter 5. Therefore the protocol is as secure as before, provided that the smart card cannot be compromised.

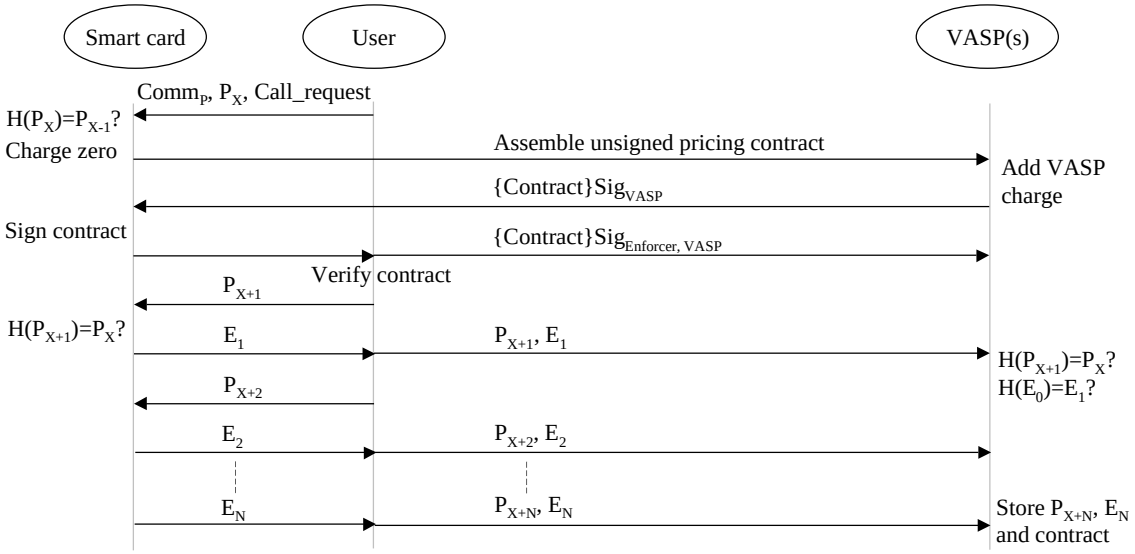


Figure 7-4 Local Payment using Smart Card Enforcer

If the smart card is broken, the user obtains the powers of the enforcer and may double spend. However, the security analysis in Chapter 5 showed that all enforcer overspending can not only be detected but also proved. It can be dealt with in the same way, by revoking the enforcer certificate and expelling the smart card holder. In a sense, a card enforcer certificate is now a special form of user certificate, something we deliberately tried to avoid initially. Nevertheless, there is no reason why groups of cards cannot share the same keys and certificate as in the previous section. This creates a distributed enforcer entity.

By combining a smart card with our scheme, we have constructed a method to allow efficient offline local party payments with a single user chain for all payments. At a high level, the smart card enforcer functionality is similar to Chaum's e-cash observer, discussed in Chapter 3, which prevents user cheating and is required for a payment to occur. A card enforcer still allows efficient multi-party micropayments and can be applied not only to local payments but all our original payment scenarios involving multiple entities.

7.4 User-to-User Payments

There has been a recent surge in interest in providing user-to-user electronic payments over the Internet [Pat00]. While both electronic cash and electronic cheques allow this, they are not the models being adopted. Instead, a central account-based model is gaining widespread use. SSL is used to securely authenticate an account holder to the online bank and payment is made by transferring funds to a payee account at the same bank. Although the payee receives an e-mail notifying them of the transfer, SSL must be used to securely verify online with the bank that payment has been made. We mentioned such schemes in Chapter 3, of which PayPal and Yahoo PayDirect are the most popular.

Since there is clearly a demand for user-to-user payments we now describe how they can be facilitated in our scheme. The solution is more flexible than current models as payment is offline and users need not share the same bank/broker. In turn, true offline operation allows more practical applications. For example, a user could pay a friend €10.00 for her share of the lunch bill, using a wireless Bluetooth link between their PDAs. The user is not restricted to local payments as the payee could equally well be remotely reachable across several networks. In addition, if the user is unreachable or roaming in distant lands, payment can be made directly into their account at any broker.

As before, the user holds monetary value which must be spent through the enforcer. For payment to another local user, the most flexible option is to use an enforcer smart card, as described in Section 7.3.1. For remote

user payments a network enforcer may be used. The first concern is how to identify the user being paid in the pricing contract. One option is to issue the payee with a certificate allowing her to sign the pricing contract as an ordinary SP does. This proves to the payer that the payee is present and is really who they claim to be. However, we want to avoid user certificates, for reasons discussed in Chapter 5, and proof of identity is usually not necessary for face-to-face payments.

A second option arises from realising that a payee must redeem value offline from a broker and that they are likely repeatedly to use the same broker with whom they have an account. Therefore, we can use a payee identity such as *user@broker* in the pricing contract. Payment hashes can then only be redeemed by depositing them into this broker account. A regular SP may cash in tokens with any broker on a per-call basis, hence the request for an SP signature during redemption. However, identification is only necessary for broker withdrawals in the case of payee accounts. As the pricing contract is assembled, the payee user identity is placed in the SP field, and the payee broker is set as the redeeming broker. Unlike a call with unknown SPs, we are leveraging the fact that the payer has the correct payee account identifier to remove the need for a payee signature.

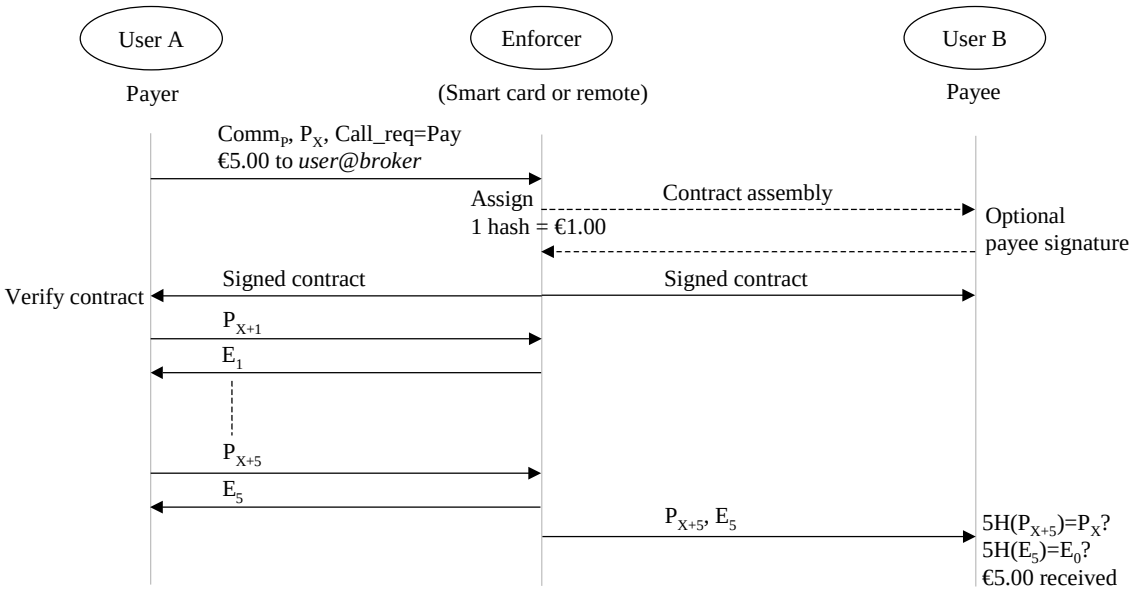


Figure 7-5 User-to-User Payment

For ongoing payments to another user, the protocol operates as before but with a payee identity in the contract. Such a scenario might arise if the payer wants to pay as a resource is consumed, such as using a friend's printer. Another example might be ongoing wagers in a game of cards. More usually, the payment is for a small lump sum. The amount could be placed in the payee's charge field in the contract and payment made by releasing a single hash. However, as discussed below, it is more secure to pay the total amount using multiple payment hashes, even if they are sent all at once, as shown in Figure 7-5. Either the payer or payee can fix the amount in the contract before it is signed by the enforcer. If the payer chooses, then the call request becomes a request to pay the payee a specific amount. If the payee does not have a certificate and is not specifying the payment amount, then steps one and two of the contract assembly need not be passed to her. Payment will not be made until the payer sees the enforcer-signed contract with the agreed amount. For remote user payments, any intermediate SPs who decide to charge for the network connection can be included in the contract and paid as before.

If a payee is not present locally or remotely, payment can be made directly into their broker account. A connection is made to the broker and a contract assembled with the broker listed as an SP. The broker signs

the contract as an SP and need not charge anything in the contract. The payee, user@broker, is listed as the final SP in the contract, and payment is made. The broker redeems the payee part of the contract, placing the funds in the listed account. The broker signature on the contract along with the endorsement hash is proof that payment was made, and can be used as a receipt. If an attacker prevents payment reaching the broker, the receipt will serve as payment re-transmission when presented to the broker.

We now consider the security implications of setting the payment hash value to be high with a maximum value equal to the total commitment worth. Provided that there is a timeout period associated with the contract, the value cannot be stolen by an attacker, as described in Section 5.9.1. The user cannot release the next token until a contract with incorrect amounts has expired. We showed that unprovable enforcer fraud is limited to stealing one hash, which is now a concern if that hash is worth the entire chain value. Enforcer cheating is not a problem if an enforcer smart card is used, unlike a remote network enforcer. For added security, a maximum value, of say €1.00, is associated with each payment hash. The user will not release the next payment until she gets the previous endorsement, which will prove any fraud that occurs. Therefore, to pay another user €5.00, the payer will need to release 5 payment hashes, and receive 5 endorsement hashes, one at a time. These messages need only pass between the payer and the enforcer, with only the highest hashes, P_5 and E_5, being sent to the payee as shown in Figure 7-5.

We have shown how the multi-party scheme can be used to allow both local and remote user-to-user payments in a more flexible and efficient manner than existing solutions on the Internet.

7.5 Mobile Ad-Hoc Network Payments

A *mobile ad-hoc network* is a group of mobile hosts which connect to each other dynamically to form a multi-hop wireless network topology. Each mobile node may act as a router for other nodes and, as they move, the routers between nodes may change. An ad-hoc network may be connected through one or more hosts to the fixed network. A detailed introduction to ad-hoc networks and their characteristics can be found in [CM99, CMC99].

In Chapter 1 we envisioned a scenario in which every mobile user held a device capable of communicating using one or more wireless technologies, such as Bluetooth, HomeRF, and wireless LAN protocols. If a user is out of range of a nearby base station then that user might be able to relay its traffic through one or more other mobile nodes to the fixed network, thereby forming an ad-hoc network. Such ad-hoc networks will have scarce and variable bandwidth. The nature of mobile nodes means that they must rely on energy-constrained operation, and relaying another entity's traffic will deplete their energy store. Under such conditions it is unlikely that mobile nodes will relay a stranger's traffic for free. Therefore, there needs to be a method to dynamically pay the multiple parties involved in providing such a service. If an entity can be paid, they are more likely to relay traffic and allow another entity to receive a better QoS than they reserve for themselves.

Before investigating if our multi-party payment scheme can be applied in an ad-hoc network, we must consider how it differs from the traditional mobile network. Several significant differences make the problem of payment much harder. In the mobile network scenario every party being paid was in the fixed network, while every party being paid in an ad-hoc network is just an ordinary mobile user. When a call is placed, the route taken through the ad-hoc network is not known in advance. However, our payment scheme requires that a pre-decided enforcer be present in the call route. Our offline enforcer selection might be able to solve this.

As mobile nodes move, the ad-hoc route may also change, due to changes in network topology and node connectivity. The enforcer, who must be present through the call, may move off the route. Each time this

occurs another pricing contract with a new enforcer and chain needs to be established. The moving nodes cause the equivalent of frequent inter-operator handover in mobile networks, a problem addressed in Section 7.9.

The routers or SPs in a call are now just regular users relaying traffic. However, our scheme relies on a trust hierarchy, shown in Figure 5-2, where enforcers are more trusted than SPs which in turn are more trusted than users. We do not want to endow enforcer privileges on regular users and yet every router may only be operated by such users. In addition, each SP on the call route was issued with a certificate that was used to verify the digital signature on the contract. Yet we want to avoid using user certificates, even though they will effectively be SPs on an ad-hoc call route.

The smart card enforcer scheme, described in Section 7.3.1, provides a good initial solution for the ad-hoc scenario. The enforcer is on the card and omnipresent with the user. This solves the problem of an enforcer roaming away, with a new chain required each time. It also ensures that enforcer privileges are not bestowed upon regular users in the call.

As part of the performance optimisations in Chapter 5, we showed how ordinary SP signatures could be removed from the contract without jeopardising security. We apply this technique to the smart card enforcer scheme so that user signatures and certificates are not required. In Section 7.4, where we considered user-to-user payments, we showed how a normal user could be paid by placing a user account number at a specific broker in the contract. Combining these two elements yields a practical solution without need for user certificates and limiting payment to named accounts only.

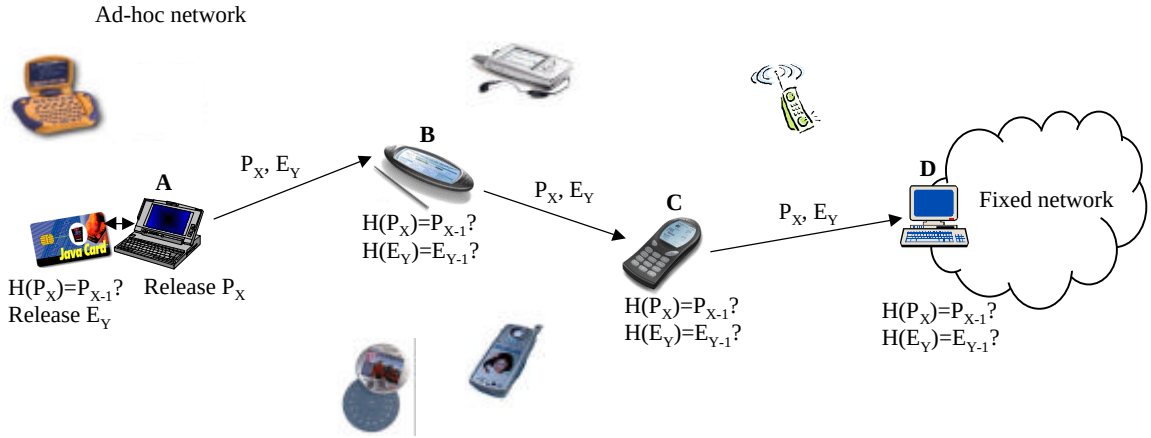


Figure 7-6 Mobile Ad-Hoc Payments using Smart Card Enforcer

Contract assembly for ad-hoc payments occurs as with the smart card enforcer protocol except that each user on the route includes their tariff details and broker account identifier. Only the enforcer signs the contract, on the smart card. Payment is ongoing with tokens being released by the user and card, as shown in Figure 7-6. Each entity will only relay traffic provided they receive valid tokens as part of a pricing contract with their tariff and account ID present.

The entity D on the edge between the ad-hoc network and the fixed network may be an ordinary SP rather than just a user. An alternative to requiring a smart card is for this entity to be the enforcer. However, since there may be multiple routes into the fixed network from a roaming mobile, this is not as flexible. It is likely that the user call will pass through several entities in the fixed network as before. While these may be included in the same contract as for the ad-hoc network, due to ad-hoc route changes it is more efficient to use a separate contract. Therefore, if entities B or C are replaced on the route, a new ad-hoc contract must be

assembled, but the same fixed network contract remains. Due to SP signatures, the fixed network contract requires more computation to generate. Either the user can pay the fixed network entities directly or she can pay entity D the total, who will then assemble the fixed contract and pay the fixed entities indirectly. The latter option, which is well suited for traffic aggregation, is discussed in Section 7.7. Traffic aggregation, where several individual flows are combined into a single flow to reduce per-flow state, is also likely at nodes near the edge of the ad-hoc network such as either C or D in Figure 7-6. Entity C may receive payment from several different nodes relaying traffic through him into the fixed network. Payment aggregation is discussed in Section 7.8.

A new ad-hoc pricing contract is required when the call route changes, but the same enforcer chain can be re-used. Contract assembly overhead is a lot less than before, as the single signature overhead is borne by the user, on her smart card, rather than by every entity in the call route. We now briefly investigate how to decrease the probability of needing to assemble further contracts mid-call. Some ad-hoc nodes will be more mobile than others. For example, the mobility, over time, of a laptop sitting in the owner's home is likely to be less than the PDA whose owner is eating at a restaurant table, which in turn is less than the mobility of a jogger's mobile phone. A device can be *mobile-aware*, in that it knows how frequently and how far it has moved, by using GPS or based on the base-station beacons received. Based on this, we suggest that a mobility profile and a *probability of movement* factor, relative to other ad-hoc mobiles, can be calculated. This probability will be very low for the home laptop, but much higher for the jogger's phone.

When a paying user selects an ad-hoc route, it could be restricted only to pass through nodes which are not likely to move off the route within the next five minutes. The caller's mobile node will also have access to the profiles and length of all previous calls and can use this information with the call request to choose an appropriate route. For example, if the call is only likely to last 10 seconds, then nodes which are not likely to move in the next minute could be used. Using these ideas, the number of new inter-call pricing contracts can be minimised.

7.6 Asynchronous Multi-Party Micropayments

Our multi-party micropayment scheme is *synchronous* since all parties are paid at the same time with the same payment hash. The value of the payment hash for each entity is set in the pricing contract. Synchronous payment assumes that all entities are using the same basic charging unit, be that volume based, time based or otherwise. However, one or more of the call entities may use a different charging unit than the others and therefore will need to receive payment at different intervals.

Such an example is a connection across multiple networks to obtain premium content from a Web server. Each NO might charge based on the volume of traffic transported while the content provider might charge per Web page, regardless of its size. Similarly, an online game provider might charge per minute of game play while the NOs may charge for the volume of traffic and QoS. We now discuss how our scheme can be adapted to allow asynchronous payments.

The easiest solution is to use a separate payment chain and pricing contract for each group in a call that uses a different charging mechanism. For example, one chain would be used to pay those who charge by volume and another to pay those who charge by content, as in our Web scenario. Each chain need not use the same enforcer. However, the user will have the extra overhead of obtaining a second payment chain for the call, storing it, and verifying two pricing contracts instead of one. If a single enforcer is used, he will have to perform double the number of digital signatures. In addition, a complete record of the call is no longer available because it has been split across two pricing contracts.

7.6.1 Dynamic Entity-Specific Payment Chains

An alternative solution that is more efficient is to include multiple hash chains in a single payment commitment. During each call, one of the chains is assigned to each group of entities using the same charging mechanism. Three chains are sufficient, as it is unlikely that there will be more than three different charging schemes within a single call. Before commitment purchase, the user needs to generate an additional two chains, although since there are now more chains they do not need to be as long as the original single chain. The user can derive all three chains from a single secret value, using the same technique as with UOBT, described in Section 3.3.3.5. We identify each of the three payment chains in the commitment using the labels PA, PB, and PC.

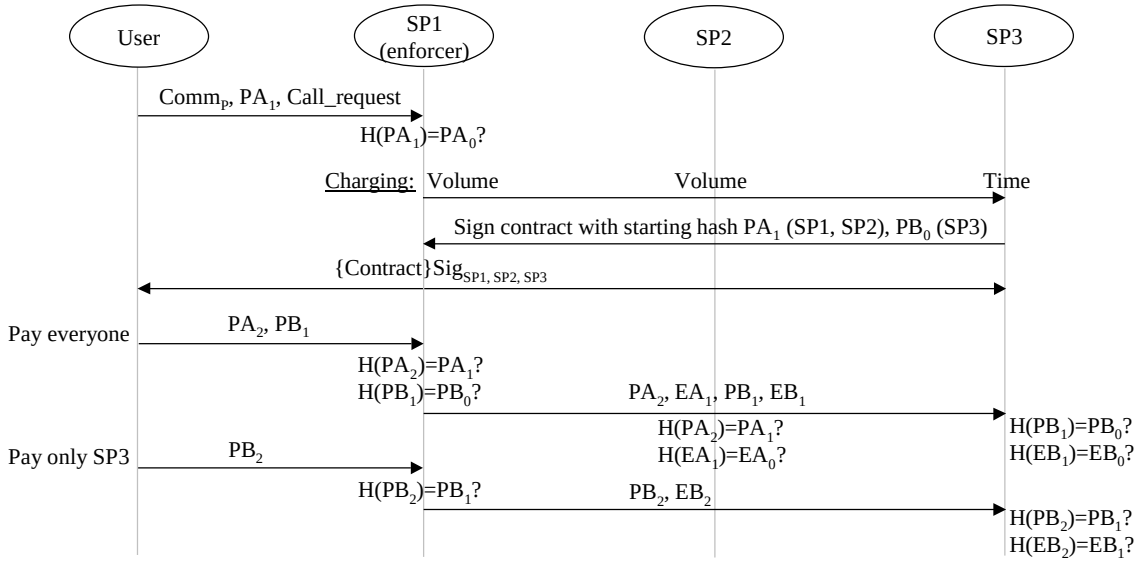


Figure 7-7 Asynchronous Payment to Two Different Charging Groups

For a new call a payment hash from the first chain, PA, is used to authenticate to the enforcer, as shown in Figure 7-7. A pricing contract is assembled using the three-way handshake as before. The charging field in the pricing contract specifies which chain is used to pay each SP, in addition to their charges. In Figure 7-7, SP1 and SP2 charge based on volume while SP3 charges based on time. As each SP adds a charging rate to the contract they compare their charging mechanism to that of the previous SPs and, if none match, they will require payment from a separate chain. This will result in the PA chain being used for volume and the PB chain used to pay for time:

Chain PA: SP1, SP2
Chain PB: SP3

This information will be efficiently encoded into the charging field of the pricing contract. For each payment chain to be used, the enforcer must commit to a corresponding endorsement chain in the pricing contract. We label the endorsement chains EA and EB to match the corresponding payment chains PA and PB. In Figure 7-7, since two payment chains are required, two endorsement chains will also be employed to prevent double redeeming as before.

During the call, the user releases payment hashes from the appropriate chain to pay each entity linked to that chain in the contract. The enforcer verifies all payment hashes and adds an endorsement hash from the corresponding chain. In Figure 7-7 the user starts by releasing a hash from each chain to pay all parties. In the second payment, just a hash from the PB chain is released so that only SP3 receives payment.

Asynchronous payment has an additional computational cost over the synchronous scheme due to having to process extra payment and endorsement chains. However, these are only lightweight hash operations and no extra signatures are required. Each ordinary SP has no extra overhead as they need only verify two hashes per payment as before. In terms of storage, the enforcer needs to record the last spent hash in each extra chain, an extra 40 bytes in total. User storage is increased by 40 bytes due to the chain anchors in the commitment. They can be replaced with a hash of the three chain anchors, rather than the anchors themselves, resulting in no storage increase for the user. Therefore the additional overhead is minimal and significantly less than using separate commitments, additional signatures, and different contracts by paying each entity separately using the synchronous scheme. We have adapted our multi-party payments to allow efficient asynchronous payments to multiple charging groups.

7.6.2 Dynamic Entity Specific Payment Tokens

In the previous section, we used multiple payment chains and made each chain specific to a group of entities. We now investigate an alternative approach where each individual payment token, a chain hash value, is made specific to an entity. This is performed by dynamically linking an identity to the token. When released, any party can verify to whom the token has been assigned, and it can only be redeemed by that entity. Such a token cannot be stolen as it passes through other SPs and is seen by eavesdroppers.

The problem is how the user can dynamically link an identity to the token. One could digitally sign both the token and the desired identity, but public key signatures are too inefficient. A one-time signature can perform the same task but to sign a 128-bit identity requires 225 hashes on average, as discussed in Section 3.3.3.6, which is still too computationally heavyweight for every payment. We leverage the fact that there are only ever a small number of SPs involved in a call, and they are identified in the pricing contract. Each identity, or group of identities arranged in a charging group, can be represented by a small number of bits. If we assume a maximum of three different charging mechanisms per call, and hence three charging groups as in the last section, each can be signed by a one-time signature using only two hashes. A one-time signature of three possible values can be represented with two hash chains, each of length two. A similar idea was used in Section 7.2 where a small one-time signature was used to select an enforcer number. Instead here we are selecting a charging group identity, GID, where each SP using the same charging mechanism is in the same group in the pricing contract.

Figure 7-8 shows both an abstract and a detailed view of a payment chain structure where a payee identity can be dynamically linked to each payment token. A mother chain, M, is used to derive all the signature chains and the payment chain itself. Two different one-time signature chains, labelled SA and SB are associated with each payment token. The two chains are used to sign the identity of the payee and the signature will comprise of one value from each chain. The final hashes of the signature chains are combined to form a single signature anchor labelled S. The signature is linked to a payment token by combining S and the payment token P together when constructing the payment hash chain. The root of the payment hash chain is derived from the mother chain M.

A fixed value can be linked to a payment hash in a chain by hashing that value with the payment hash to form the next payment hash. Figure 7-8 shows the details of how the payment chain is constructed with a one-time signature linked to each payment hash. The entire structure is derived from a single value, M_{2N+1}, held secret by the user. The length of the one time signature chains, labelled SA and SB, is chosen to be long enough to sign the maximum possible number of different identities. In this case, we use a maximum of three group identities, and therefore use chains of length two. The SA chain represents the GID, and the SB chain provides a checksum, as explained in Chapter 3.

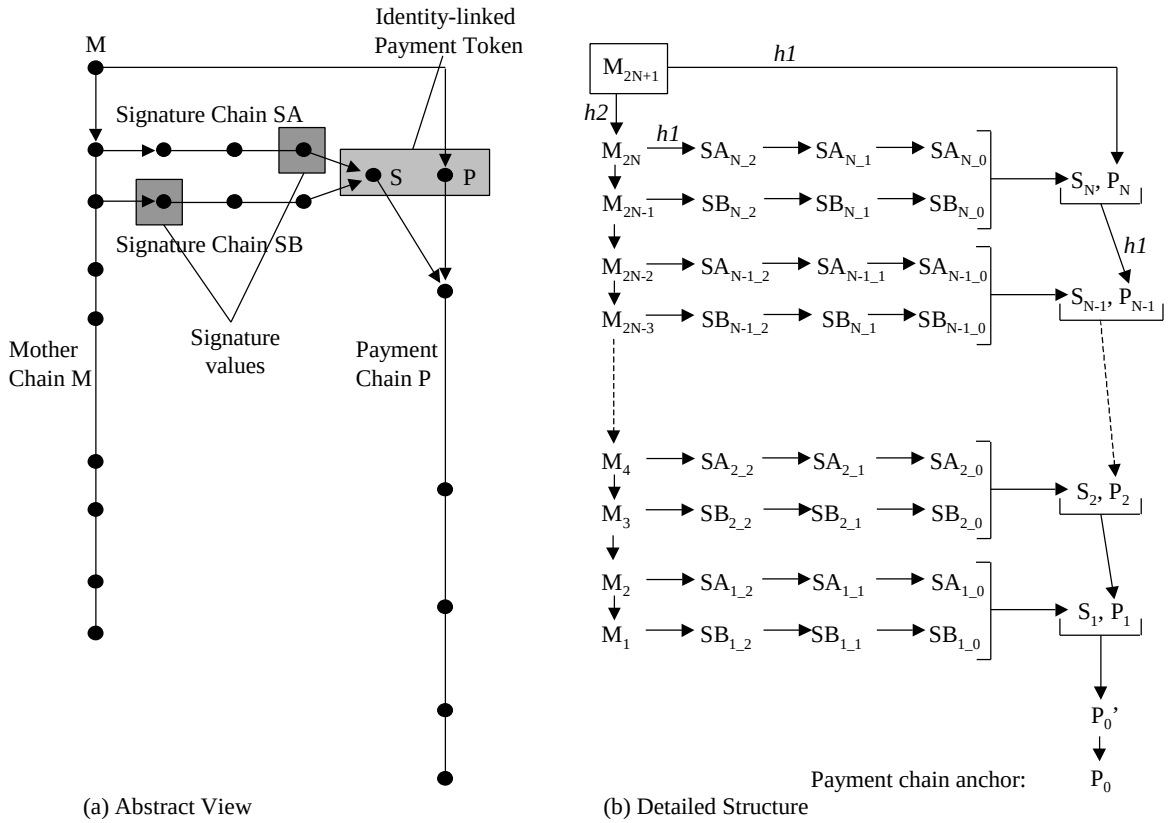


Figure 7-8 Entity Specific Payment Tokens using One-Time Signatures

The *M* chain of length *2N+1* is generated from the user secret M_{2N+1}, using a hash function labelled *h2*. The hashes from this chain are grouped into pairs, from which the SA and SB signature chains are generated, using a different hash function *h1*. The public anchors of each one-time signature are SA_{X_0}, SB_{X_0} where *X* is the number of the corresponding payment hash. A payment hash P_2 could be linked to the signature by concatenating and hashing the SA_{2_0}, SB_{2_0}, and P_2 values. This would require storing the two signature anchors just to derive the next payment hash. To halve this storage cost we instead take a hash of the signature anchors to form S_2, and link that with the payment hash as shown.

As before, the payment chain anchor P_0 is signed by the broker as part of the payment commitment. The first payment hash, P_0', is not linked with a signature as it is always used just to authenticate the call request, as described in Chapter 5.

Figure 7-9 shows a call being made with group-specific tokens. The first call request for a new chain is authenticated with P_0' but, for later calls, an unsigned token (S_X, P_X) is used. In the pricing contract, the SPs form groups identified by a GID based on their charging mechanism. The user makes payments specific to one GID, by using the one-time signature linked to the token. In this case, a token signed for GID#0 can be redeemed by SP1 and SP2, while a token signed for GID#1 can only be redeemed by SP3.

The first payment made is for GID#0 only. The user releases P_1, SA_{1_0} representing GID#0, and SB_{1_2} as the checksum count of the unreleased parts of the SA chain. The enforcer verifies the signature to obtain S_1 and verifies the payment hash. The enforcer releases a single endorsement hash. There is only a single endorsement chain used, unlike the multiple chains scenario.

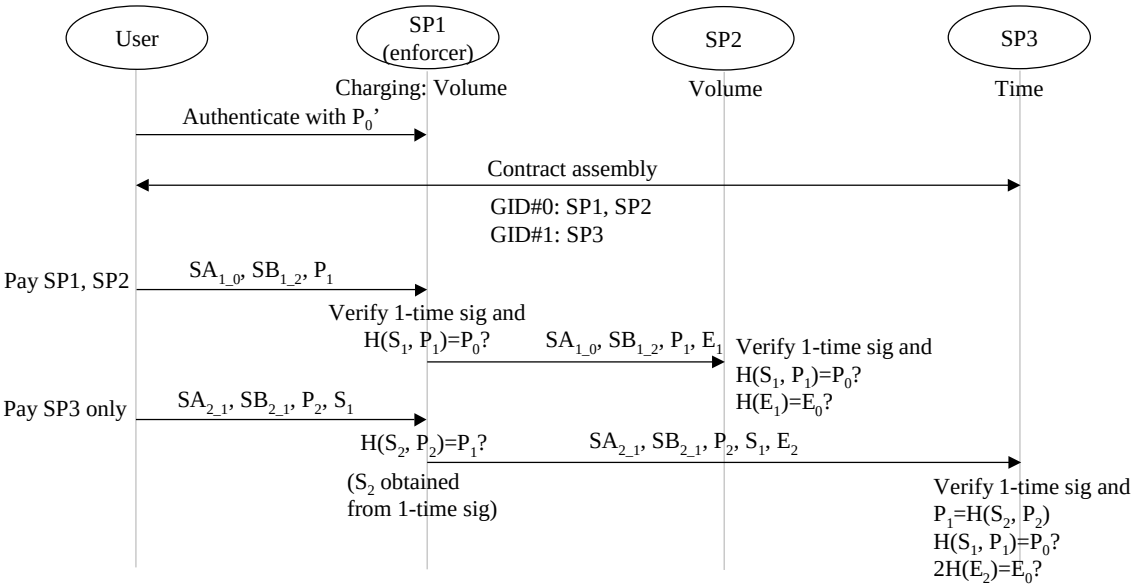


Figure 7-9 Asynchronous Payment with Entity Specific Tokens

The signed token is passed onto SP2 who verifies the signature, the token, and the endorsement hash. Each paid party must record the signature hashes, the payment hash, and the endorsement hash. Since the token is not specific to SP3, there is no need to forward it onto him. Even if SP3 does obtain the token he cannot gain from it.

The second payment is for SP3 only, and hence is signed for GID#1 using SA_{2_1} and SB_{2_1}. The payment hash P_2 needs to be verified by the enforcer before he releases a matching endorsement hash. This is performed by obtaining S_2 from the signature, requiring three hashes, and hashing it with P_2 to obtain the last payment P_1.

The payee SP3 needs to verify that P_2 and hence P_1 belong to the P_0 payment chain. S_1 is sent along in order to allow P_1 to be verified. Alternatively, S_1 can be obtained and stored from the first payment to GID#0, but it is simpler not to involve an SP in the processing of an earlier token not meant for him. In an actual implementation, indices and identifiers, totalling a few bytes in length, will be included to distinguish the hash values from one another.

We have proposed entity-specific one-time signature tokens as an alternative means of performing asynchronous payments. Unlike the previous solution, they require only a single payment and endorsement chain. Generation and verification of the one-time signature requires three hashes or fewer. No extra storage is required by the user. The drawback of the scheme is the extra storage imposed on the SPs. Each payee must store the signature hashes SA_X, SB_X for each payment. In addition, for each earlier token assigned to another group, the signature link S_X must be stored. This is necessary to verify the corresponding payment token P_X. While the P_X values can be obtained by hashing tokens higher up the chain, the S_X value cannot. Each S_X value is already a hash of the signature chains SA_{X_0}, SB_{X_0}. It therefore cannot be obtained from another S_X using a hash function, as this would require a hash function collision. The storage requirements of one-time signature tokens remains less than MicroMint, where each coin, consisting of four pre-images, must be remembered. In addition to using the one-time signature to encode payee identity, it can also encode the value of the token, rather than fixing it in the contract.

7.7 Indirect Multi-Party Payments

In our multi-party payment solution, the user pays all entities directly by transmitting the same single token to all parties. An alternative model which has the same net result is for each entity to only directly pay their downstream party. We call this process *indirect multi-party payments*.

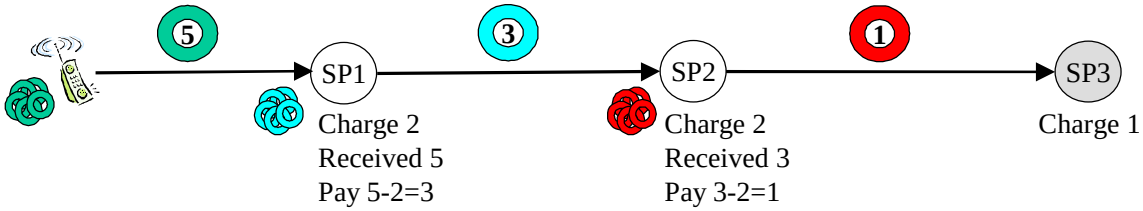


Figure 7-10 Indirect Multi-Party Payment via Downstream Entity Payments

The mobile user pays the aggregated call cost per charging unit from all entities involved in the call, to the local network operator. That NO then makes a payment, equal to the total received payment minus the amount he is due, to the next entity downstream in the connection. The NO who receives this payment does the same for his downstream entity and so on until the final entity gets paid. The largest payment will be from the mobile to SP1, with decreasing amounts being paid further down the route. For example, consider a call involving service providers as shown in Figure 7-10. The mobile might pay SP1 5 cents per unit time, of which 3 cents might be paid to SP2, who in turn pays SP3 1 cent. Thus SP1 charges 2 cents, SP2 charges 2 cents, and SP3 charges 1 cent per unit time for providing their part of the call.

Since each SP is paying the entity downstream, we can leverage this flexibility for use in a *migration scenario* where some SPs still use traditional CDR billing. A connected neighbouring SP, present in the call route, can accept payment on behalf of a legacy SP who still generates CDRs and bills other SPs. Another important scenario is when multiple simultaneous calls take place between two SPs. Instead of making a number of separate payments, a single *aggregated payment* could be made, as described in Section 7.8. Both of these scenarios provide reason alone for examining the feasibility of indirect multi-party payments. We now investigate how our original scheme can be tailored for such indirect multi-party payments.

In our original direct multi-party scheme, payment chains were purchased for a specific enforcer, through which they were spent. In the indirect case, the user pays only her local SP and so payment chains will be purchased for a specific local SP. The chain purchase protocol remains the same as before, except the enforcer identity is replaced with a local NO identity. Indeed, our solution allowing offline selection of an enforcer, in Section 7.2, can also be used to choose the local SP offline. In this way, the mobile does not need to know which particular mobile network it will be in when buying chains.

With indirect payment, chains are vendor specific and their entire value is spent at a single SP. Since the number of SPs is greater than the number of enforcers previously, we place less trust in them. Therefore we do not allow every SP to fix the value per payment hash dynamically, but instead fix the value in the payment commitment during purchase. The value per hash is equal to the total chain value, divided by the length, and will typically be a small value, such as €0.01. Fixed value tokens prevent the SP from stealing them.

Each SP pays their downstream entity with tokens from a *different payment chain* during a call. For example, in Figure 7-12, the mobile user pays SP1 using payment hash chain P, while SP1 pays SP2 using payment chain Q, and SP2 pays SP3 using payment chain R. Typically, each SP will know to which other SPs it is directly connected and can have chains ready in advance. Where two SPs trust each other, credit hashes,

generated by the SP instead of prepaid ones, can be used. A long payment chain between two SPs can be used for many calls.

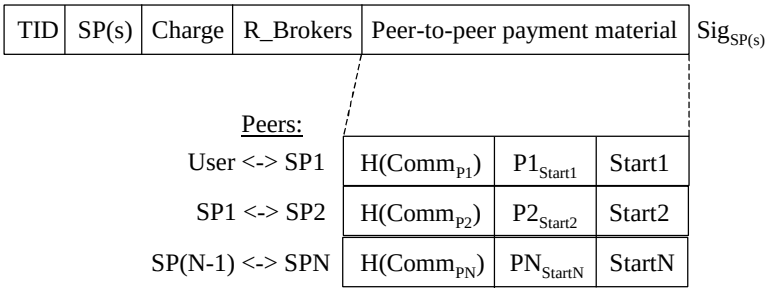


Figure 7-11 Pricing Contract for Indirect Multi-Party Payment

To make a call, a modified pricing contract is assembled using the three-way handshake as before. The elements sent by the user to the first SP remain unchanged. However, some fields of the pricing contract differ, as shown in Figure 7-11. The TID, SP identities, charge, and redeeming brokers field are constructed as before. However, there is now no endorsement commitment or hash value fields. Instead, the payment chain details used between each pair of entities is included. Each party provides the downstream entity with the payment commitment $Comm_{PX}$ which is to be used to pay them, the starting payment hash, P_{StartX}, and its position in the chain, along with the partially constructed contract.

Each payment commitment is specific to one downstream SP, and is not needed by the other SPs. Therefore, rather than including a number of entire 157-byte payment commitments in the contract, only a hash of each is included. For calls involving 3 entities or fewer this yields a smaller pricing contract than the original scheme. The last spent hash and its index are included as the starting anchor for this call.

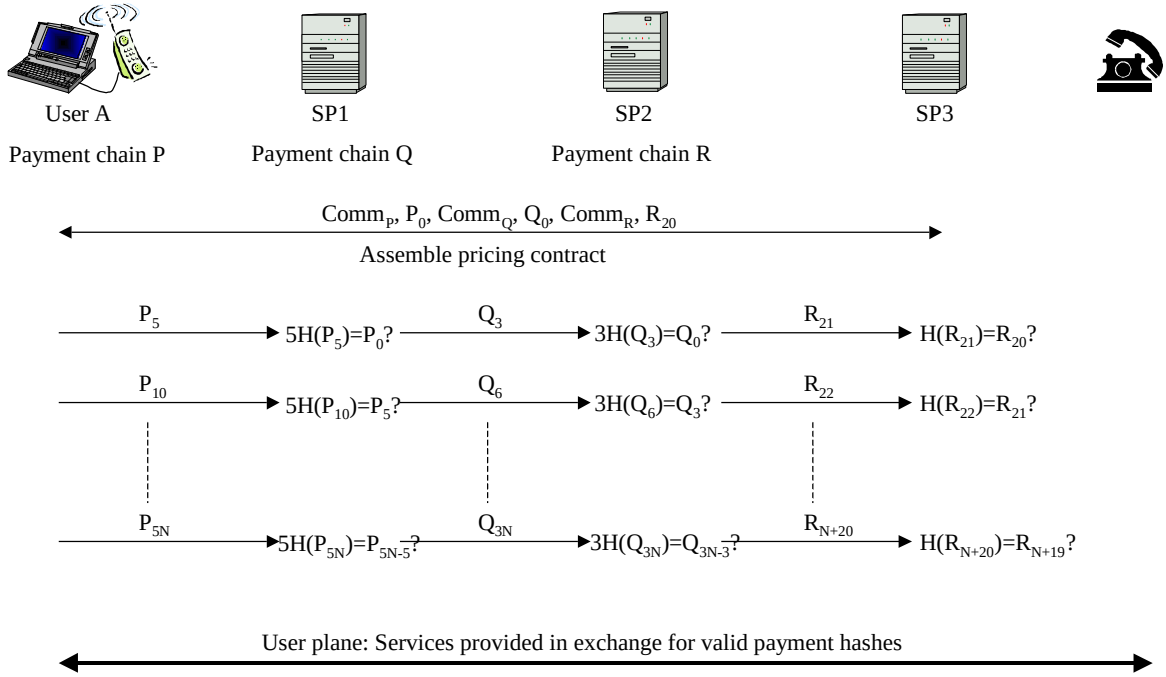


Figure 7-12 Indirect Multi-Party Payment using Multiple Chains

Having agreed to the pricing contract, the user begins the call by releasing a payment hash worth the total amount due per charging unit. For example, in Figure 7-12 SP1, SP2 and SP3 might each charge 2, 2 and 1 cents per unit respectively for the call, yielding a total charge of 5 cents. If $Comm_P$ represents a new payment chain with hash values worth 1 cent, then P_5, the fifth hash in the chain, is sent to SP1. SP1 applies the hash function five times to this payment hash and, if the payment is valid, the result will be the final hash P_0. The value of a payment hash is fixed in the broker commitment. By sending P_5 after P_0 it is the equivalent of sending all five payment hashes, P_1 to P_5, since they can be obtained from P_5 by repeatedly hashing.

In turn, SP1 pays SP2 using payment hashes worth 1 cent from the chain defined by $Comm_Q$. SP1 pays SP2 the remaining 3 cents by sending Q_3. SP2 applies three repeated hash functions to Q_3 to obtain Q_0, proving the payment is valid. SP2 starts to pay SP3 with an unspent hash R_{21} from a partly used chain with commitment $Comm_R$. In Figure 7-12, N is the number of payments made.

At the end of the day, each SP redeems tokens as before by sending the pricing contract, and the highest payment token and its index to their preferred broker:

$$Redeem = \{Pricing\_contract, P_X, X\}Sig_{SP}$$

Allowing an SP to nominate the redeeming broker in an SP signed contract will allow co-operating cheating SPs to double redeem at multiple brokers with post-fact detection. If this becomes a problem, SPs can be forced to redeem directly from the issuing broker.

The pricing contract is less important with indirect payments. Here, its purpose is to allow verifiable dynamic tariffs and charging schemes, fix the starting hash for each payment commitment and create a signed record of the call. The payment material is included in the contract in order to link tokens from part of a chain to the call route. Depending on the number of tokens that are redeemed, the duration or volume of the call along that route to the destination can be ascertained by an independent third party. However, the payment material does not need to be included in order for the SP to receive payment. The value of each token is fixed at chain purchase, and knowledge of each token is therefore proof of payment by the user to the named SP in the commitment. Value is specific to that named SP and cannot be redeemed by any others.

The charge fields signed by each SP guarantee that the SP is present and this is their correct rate at the time. The optimisation in Section 5.10 allowed SP signatures to be removed, provided the user was prepared to pay the price presented.

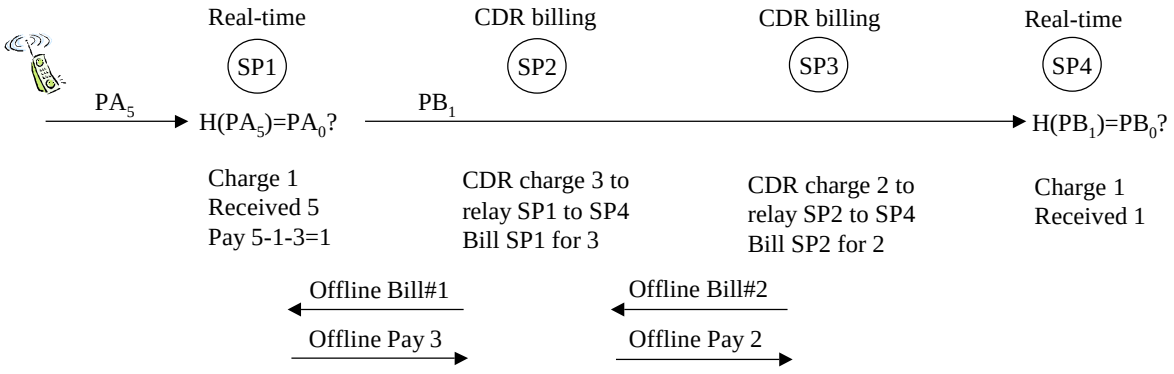


Figure 7-13 Real-Time Payment Inter-Working with Legacy CDR Billing During Migration

Therefore, under circumstances where authenticated charges and a signed record of the call are not required, the signed pricing contract can be omitted altogether. Charge quotes can still be obtained from each SP using the first step of contract assembly, with each entity then paying their downstream entity. Here, call pricing is decoupled from actual payment. Indeed, removal of the pricing contract allows a more traditional model where the local mobile NO can charge the user a retail rate while paying its downstream entity a wholesale rate.

In terms of computation, each entity will have to perform more hashes per payment than previously. This is due to the need to release tokens from their own chain as well as verifying fixed value tokens. However, no signatures may be necessary at call setup as discussed above.

In the migration scenario, a legacy SP records traffic usage in CDRs and bills the interconnected SP from whom the traffic originates. That SP requires real-time payment from its upstream entity that is enough to cover its own charges as well as the CDR SP, as shown in Figure 7-13. Multiple legacy SPs can appear on the call route, provided that there is at least one SP accepting real-time payment.

7.8 Aggregated Payments in the Core Network

The topology of an internetwork usually consists of a number of *edge networks* to which users are directly connected, and a set of *core networks* through which the edge networks are interconnected. Since a core network relays traffic between all the different edge networks, it carries the greatest amount of traffic or number of calls through its routers.

There are two main service model architectures, defined by the IETF, for providing quality-of-service (QoS) on the Internet. The original architecture is based on per-flow *integrated services (IntServ)* [BCS94, Whi97]. With IntServ per-flow resource reservation takes place using the RSVP reservation protocol [BZBH+97], which requires each domain to process and maintain state on each individual flow. Our original multi-party micropayment protocol is well suited to the IntServ model as each SP keeps per-flow payment state. Indeed, the pricing contract could be integrated with the RSVP protocol so that pricing and payment information is exchanged as call resources are reserved.

However, with possibly hundreds of thousands of simultaneous flows passing through routers in the core network, processing each flow individually is not a scalable solution. The second IETF QoS architecture addresses this by grouping individual flows together to provide class-based *differentiated services (DiffServ)* [BBCD+98, MEH00]. Packets are assigned to one of a small number of priority classes, which correspond to the QoS provided at each router. State information is reduced in the core but different QoS levels are still available. We now describe how per-flow payment processing and state information used by our multi-party payment scheme can be reduced in the core network. This is done by aggregating multiple payment streams between two entities into a single payment.

Payment aggregation uses the indirect multi-party micropayment scheme of the previous section, where each entity pays the party downstream. Instead of a payment chain per user flow, a single chain per group of aggregated flows is used. In a real-world scenario, a number of calls are likely to be made to the same destination network at the same time. Some of these calls will originate in the same source network while others will originate in different edge networks. The call paths form a tree, the leaves of which are the users in the different edge networks and the root of which is the final destination network. We perform tree-based aggregation of payment streams going to the same final network as shown in Figure 7-14. Tree-based aggregation has also been used for resource reservation in BGRP[PHS00] and RNAP[WS00].

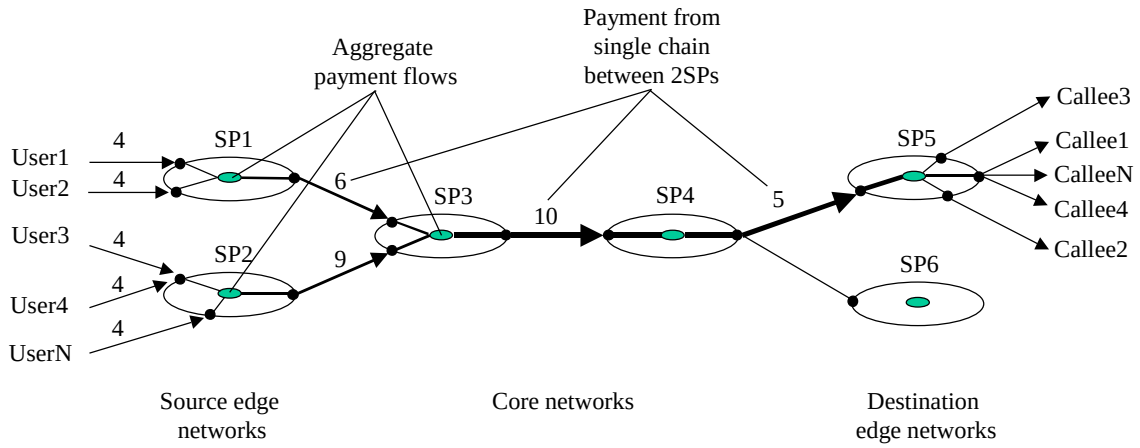


Figure 7-14 Payment Stream Aggregation

In Figure 7-14, User1 and User2 both spend tokens from separate chains with SP1. Each edge network charges 1 unit per user, and SP3 charges 2 units for traffic from SP1 and 3 units for traffic from SP2. Both SP4 and SP5 charge 5 units for the aggregated traffic. The SP1 payment processor subtracts the value it is owed from each user and pays the rest to SP3 in an aggregated payment. SP1 uses a single SP1 chain to pay SP3, who is also paid by SP2. In turn, SP3 aggregates payments from SP1 and SP2, which are both going to the destination network SP5, and makes a single payment from a single chain to SP4. There are no other calls arriving at SP4 destined for SP5 and so no further aggregation can take place. Again a payment from a single chain is made from SP4 to SP5.

The diagram shows all payment within a domain being processed centrally. Based on whether good payment is received or not, the payment processor will inform the routers in the domain as to which QoS to provide the traffic. The payment functionality can equally well be implemented at the edge routers of the domain. The ingress routers will verify payment before forwarding traffic while the egress routers will add payment as necessary for the downstream domain.

Aggregated payments result in both computational and communications savings in the core network. For example, SP4 only has to process and keep state of one token from a single hash chain, in order to be paid for the traffic originally from 5 separate calls. In the original scheme, SP4 would receive five different tokens from five different hash chains in order to be paid the same amount. The last received token in each chain would be maintained as state information. For a large number of simultaneous calls in heavily loaded core networks, aggregated payments will allow the payment system to scale.

7.8.1 Destination Networks with Low Incoming Traffic

In Chapter 1, a scenario was depicted in which anyone with wireless connectivity and a fixed network connection could provide NO services. Our main example has been a NO providing a Bluetooth connection through his PC into the fixed network. Therefore it is likely that there will be a large number of small NO's, typically providing picocell connectivity, who are not likely to receive many simultaneous incoming calls. Since our payment aggregation is based on calls to the same destination network it would not work well in this environment where many calls are going to independent picocell networks.

However, it is likely that many of these calls will pass through the same NO, perhaps the regional provider, before entering the different picocell networks. In this case, payment aggregation should take place for all calls destined for the last large NO in the route. This will allow the same performance savings as before. The picocell NO can be paid either by the upstream NO directly or by the paying user herself. For the upstream

NO to pay the picocell will require that upstream entity to have a separate independent chain for each picocell used. While feasible, this will increase the load in the core because payment deaggregation must take place and the appropriate amounts be paid to each picocell.

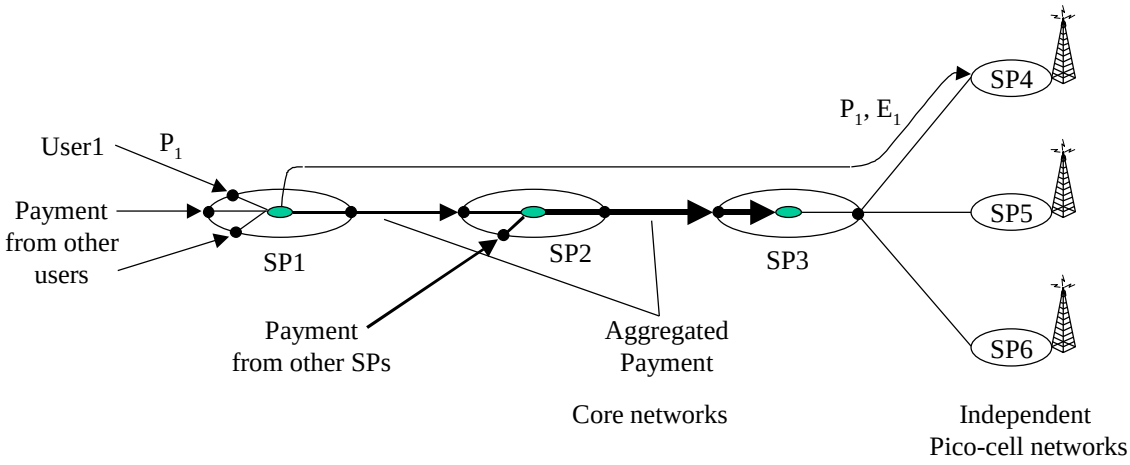


Figure 7-15 Direct and Indirect Multi-Party Payment Hybrid

A more efficient solution is to get the user to pay both the first and last NO in the call route using the original direct multi-party protocol in Chapter 5. The local NO receives and endorses tokens which are passed transparently through the core network to the picocell NO, as shown in Figure 7-15. In the diagram User1 pays SP1 and SP4 directly using a token from the same chain. SP2 and SP3 are paid from a single chain from their upstream entity. All payment streams destined for SP3 are aggregated. This removes the processing overhead from the core. In addition, either SP1 or SP4 can act as the enforcer, giving more flexibility.

For efficient aggregation, this technique should be used where calls are destined for networks which experience a low level of incoming calls. This hybrid combination of both direct and indirect multi-party payments may be more efficient than using either alone, depending on the network topology.

7.8.2 Aggregated Prices

If each SP must sign a pricing contract per call, then a per-flow overhead will be re-introduced into the core. In addition, the contract signature is the most computationally expensive part of the payment protocol. Instead, we envision an SP in the core signing an aggregated pricing contract, solely for use with its upstream SP. For example, in Figure 7-14, SP4 will establish a pricing contract with SP3 who will establish a different pricing contract with a different chain with SP1. This contract will specify a range of prices dependent on the amount of traffic and QoS required by the aggregated call stream from the upstream entity.

As calls finish and new calls begin, the volume of traffic or number of aggregated calls passed to a core SP will fluctuate. The pricing contract needs to allow for such changes so that the paying upstream SP can respond by releasing more or fewer tokens from the single payment chain used to pay for the aggregated call stream.

As an example, an SP contract might specify the rate for each megabyte of traffic passed into each different DiffServ QoS class. Depending on the traffic aggregated from a number of calls, the SP can release the appropriate number of tokens. The selection of appropriate pricing algorithms, perhaps congestion dependent, is discussed further in the future work section of Chapter 8. Depending on the traffic aggregated from a number of calls, the SP can release the appropriate number of tokens. The duration of a contract will

depend on the flexibility of the pricing specification and the length of the corresponding chain. We envision such contracts lasting for the lifetime of many calls, typically for several hours.

An SP who aggregates calls and pays the downstream SP for transporting the aggregated stream to the common destination will know how to split the aggregated cost between its upstream senders. These senders may be other SPs or users. Depending on how prices are specified in an aggregated contract, they may or may not be useful to SPs or users further upstream. Instead of presenting each of the aggregated pricing contracts along the route to the user, we envision the local SP charging the user a retail rate based on the wholesale rate of his aggregated contract with the downstream SP. The user will have a separate pricing contract per call with the local SP. In the case of a hybrid payment system, which is likely to be the more common case, the user pricing contract will include at least the first and final SP in the call route.

We have adapted our payment protocol to scale for use in core networks which handle hundreds of thousands of simultaneous calls. The approach taken was to aggregate payment in a branching tree whose root was the final large network in the call route. As the DiffServ model gains popularity, our solution provides an efficient means of inter-domain payment for different DiffServ classes with variable traffic sizes.

7.9 Payment with Frequent Inter-System Handover

Current mobile cellular systems are based on large physical cells. Each mobile operator typically controls many cells covering an entire city, region, or country. In such an environment, there is no need to handover an active call from one operator's network to another. All handoffs are intra-operator so the same NO can continue to be paid using the original pricing contract. Payment processing will take place at a central point in the operator's network, and need not be migrated as handover occurs.

However, Chapter 1 described how many independent wireless cells with a small limited range, called *picocells* can be formed using technologies such as Bluetooth, WaveLAN, HomeRF, or DECT. When the creators of such a picocell can easily be paid in real-time for their services, a large number of independent picocells may emerge in dense populated areas, providing faster and cheaper service than the large wide-area cells based on GSM or UMTS. Our multi-party payment system was designed for such a scenario where a user roams into a local picocell, pays for one or more calls, and then leaves.

It is possible that a user may wander out of the original picocell coverage before they finish their call. Therefore, it is desirable to provide inter-system handover, between different independent wireless cells, to allow ongoing calls to continue as the user moves. A number of protocols have been proposed to allow seamless mobility and handover between picocells providing IP network connectivity as part of a larger Mobile IP system. Such proposals include Cellular IP [CGKT+00], Hawaii [RPST+00] and TeleMIP [DMAD00]. While each of these schemes has considered the cluster of picocells as belonging to part of the same administrative domain, it should be possible to extend the ideas to independent picocells connected to a common fixed network. Assuming inter-system handover will become possible, we now consider how our payment system would cope with such handovers. Such a scenario might arise when a Bluetooth user wanders from one picocell to another as shown in Figure 7-16.

A handover between operators would require a new pricing contract because all SPs in the call route are present in the contract. This would need to involve all the downstream SPs, even those in distant networks. However, a key design goal of the picocell handover schemes mentioned earlier is to prevent that handover causing unnecessary signalling in a distant network. For example, in the case of cellular IP a picocell handover is not visible in the larger Mobile IP environment. Applying the same design goal to our payment

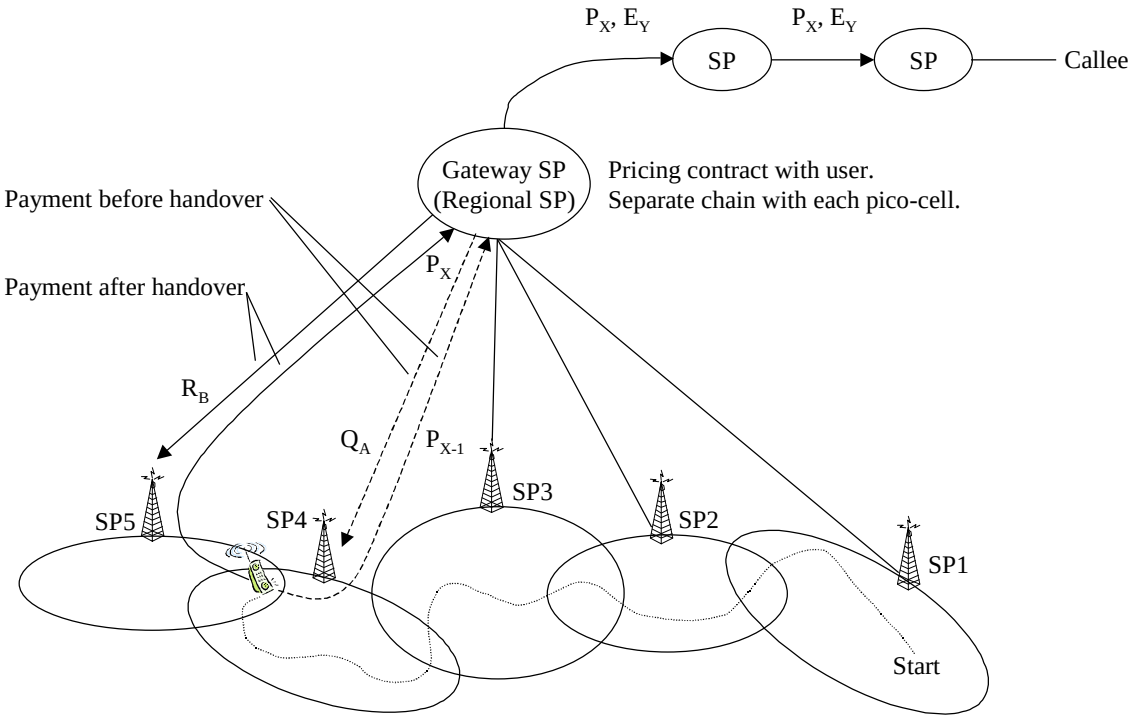


Figure 7-16 Inter-SP Handover with Ongoing Payment

system requires that a distant downstream SP, outside the local picocell environment, should not be aware of or affected by the handover. We wish to use the same pricing contract and keep paying with tokens from the same chain after handover has occurred.

The solution is not to include each individual picocell SP in the contract. Instead a *gateway SP* is included who will accept payment from the user on behalf of a picocell, and then pay that picocell directly using another chain. A cluster or group of picocell NOs need to use the same gateway SP in order to allow handover. Such a gateway SP might be the regional NO who provides fixed network access to all the pico-SPs. The advantage of this is that the regional SP will always be an SP in the call route. For calls from picocells, the gateway SP is listed as the first SP in the pricing contract.

The actual physical location of the gateway SP is not critical as long as it is reachable by each picocell NO over a fast network connection. Indeed, one of the picocell NOs themselves could act as the gateway SP for the other picocells, with user payment routed to that SP for verification, as well as being relayed to other downstream SPs.

The gateway SP charges for the picocell services and for its own services if it is on the call route. Our original multi-party scheme is used to pay the gateway SP and all other downstream SPs. In turn, the gateway SP uses a separate long-lived chain to pay the appropriate picocell in real-time. If the picocell does not receive payment, then it assumes that the gateway did not get paid by the user and terminates the user call through its base station.

Typically, each picocell will only handle a small number of simultaneous users. For instance the Bluetooth specification [Blu99] limits one master Bluetooth network card to have a maximum of eight slave connections, and therefore eight possible connected users. A single payment chain will be used to make an aggregated payment from the gateway to the cell for all the individual user payments going to the gateway. In order to be able to terminate specific calls, the gateway will therefore need to indicate to the cell which user

has stopped paying. Alternatively, the cell can monitor the user payment flows, verifying payments destined for the gateway, and will therefore know when payment stops.

There are different handover mechanisms proposed [CGKT+00], some of which drop incoming packets and others which reduce the chance of this occurring. Two simultaneous connections, one with the old picocell, and the other with the new, may be present during handover. In order to re-route incoming traffic to the new picocell, the crossover SP, which is the regional SP in Figure 7-16, will be informed of the handover status. When a handoff has successfully completed, the gateway SP stops paying the original picocell NO and starts paying the operator of the new cell, by matching user payments. The user continues to pay the gateway and all downstream SPs using the original chain. To the user payment process, the handover is seamless.

In Figure 7-16, the user moves from SP1 to SP5 during a call, passing through SP2, SP3 and SP4. The call is started in SP1 by paying with tokens, P_X, from chain P. The regional SP acts as the enforcer, adds an endorsement before passing payment downstream to other SPs involved in the call. The gateway SP, which is also the regional SP, pays each SP with tokens from a separate chain. Payment before and after the handover from SP4 to SP5 is shown. SP4 is paid with tokens from chain Q before the handover. With only one paying user in cell SP4, a token Q_A is released by the gateway for every valid user P_X token. After handover, the user is still paying for the call using tokens from chain P, but the gateway is now paying SP5 using tokens from a payment chain R.

One open issue is when the new picocell should receive its first payment. If the last user payment was made just before handover, and the picocell share of this was given to the old cell, then the new SP will not receive payment until the next payment interval. The user can be forced to make a new payment immediately after a handover. While this will satisfy the new SP, it will also mean that all the other SPs in the call route will receive an extra payment. This may seem unfair to the user but it is no different than what she would pay if she had terminated the original call and started a new one to the same destination.

The other option is just to make the new SP wait for payment, meaning that the interval while waiting is provided by him for free. This is acceptable because, in this case, the old SP received the extra payment just before handover whereas in future handovers it could be the opposite. Over time, the small differences will average out. The payment interval should be kept short to minimise the effect and the new picocell should receive an indication from the gateway SP that it is indeed a valid handover. We prefer the second option because the amounts involved are almost trivial and it costs the picocell operator almost nothing to provide a wireless link for a short time interval. This solution also suits the scenario where the user is paying by traffic volume and may pass through one or more picocells with an active connection without actually sending or receiving any traffic. Frequent hopping back and forth between two picocells can also be handled by the solution. Finally, it also prevents the user being charged for duplicate packets that are delivered through both base stations during the handover.

A second issue is that the new picocell SP will have to use the same tariffs, as specified in the pricing contract, as the original SP. The new SP can always force a new contract to be assembled as part of handover if they cannot agree to the tariff. In light of this, the picocell tariffs should be made flexible enough to handle handover to another cell offering superior or inferior bandwidth or QoS, without needing a new contract. We envisage a local group of picocell NOs agreeing to a set of flexible tariffs, with the user selecting an appropriate one for the contract. The tariffs of local picocells will be collected transparently by mobile devices as they pass through the cells making them available before the contract is assembled.

We have shown how to handle payment efficiently in an environment allowing seamless handover between independent mobile providers. The need for a new pricing contract is avoided by employing a gateway mediator to pay each picocell individually using separate chains, as in the indirect multi-party payment solution. With low-cost wireless technologies and a method to pay any provider for services in real-time, we believe that independently run picocells can provide a valuable high-speed wireless network infrastructure in any large city.

7.10 Summary

The original multi-party micropayment protocol has been extended and optimised for a number of important settings. The most practical of these was the user-to-user and local area payments. The payment chains in a mobile device can now be used as transferable money in a mobile wallet. A user can literally just point their mobile device and pay. A smart card was introduced to provide extra security, by acting as the enforcer in such payments. It was also shown how the smart card could be used to make a payment commitment specific to an enforcer some time after it had been purchased. A compromise of the smart card limits the maximum fraud to a predefined limited amount rather than allowing infinite spending ability. Asynchronous multi-party payments were proposed to allow SPs in the same call to use different charging mechanisms and receive payment at different times. An example where this is useful is content services, where intermediate SPs charge based on volume delivered, while the content provider charges based on the content itself.

The reach of traditional mobile networks was extended by showing how payments could be made to relay traffic across ad-hoc networks. In cellular mobile networks, a technique to allow payment to continue seamlessly in the presence of inter-SP handover was introduced. To provide scalability in the core network, payments can be aggregated along a route so that heavily loaded SPs are paid once for a number of calls rather then individually for each. Such aggregation can be used to pay efficiently for DiffServ traffic between operators.

Multi-party micropayments have been extended to be useful in a much wider range of settings than just as a replacement for traditional mobile billing systems. Payments can be made to anybody who provides a service from a mobile device in any location, paving the way for ubiquitous mobile payments where money constantly flows through every inter-connected device. In the final chapter, we highlight the contributions of the multi-party payment protocols and discuss further work that remains.

8 Conclusions

"Its all over, and its all about to begin"
From "Philadelphia, Here I Come!", by Brian Friel.

8.1 Summary of Contributions

The main goal of this thesis was to investigate the use of electronic payment techniques as a replacement for traditional billing in a mobile communications environment. We have proposed the first multi-party micropayment protocol. It solves the problems that existing and future mobile billing systems will experience in an environment of ubiquitous mobile communications.

Our review of billing techniques highlighted the inadequacies of current billing and the emerging problems when applied to the realm of large-scale packet networks with QoS and a huge variety of applications. Every state change and content request must be captured to allow tariffs to be applied accurately. In addition, every service provider must be trusted to generate accurate usage records, an unacceptable requirement with ubiquitous communications where any party can provide services. These problems provided the motivation for our research and our multi-party payment scheme was designed as a secure, efficient, flexible solution.

Micropayments emerged in 1995 as a means of making efficient repeated electronic payments of small amounts. We have contributed what we believe to be the first extensive survey of the micropayment literature. A taxonomy was created based on the cryptographic techniques used. The properties of each of six groups were compared and contrasted and differences between individual schemes within each group were compared. Hash chain schemes were shown to have received the most attention in the literature, and are the most fault tolerant if payment is lost in transit. The efficiency of many schemes was shown to rely on user trust with post-fact detection, a condition we deem to be impractical to enforce in a global mobile environment.

Remaining open issues in micropayment research were discussed. Based on observations made during the survey we put forward three new micropayment contributions. We proposed a method to extend hash chains infinitely using one-time signatures, a shared secret key scheme requiring no vendor trust, and an efficient method to derive and store hash chain trees.

The performance of electronic payments and micropayments depends largely on the underlying cryptographic algorithms and constructs used. By benchmarking specific algorithms on commodity platforms and with popular implementation languages, we have provided a means to estimate accurately the performance of systems which make frequent use of these algorithms. These measurements can be applied not only to payment systems but to any network security system in general.

We applied the cryptographic benchmarks to gain a clear understanding of how micropayment schemes compared in terms of computation, communication, and storage. The performance of representative macropayment schemes was also considered. The poor performance of schemes relying on public key techniques and online connections was confirmed. Hash chain schemes yielded the minimum vendor computation for ongoing payments and the lowest bandwidth requirements. The micropayment performance charts illustrate how design decision tradeoffs result in varying performance abilities.

A major contribution of this dissertation was the multi-party micropayment protocol, which we believe to be the first of its kind. Micropayment research has exclusively focused on the problem of paying one party while our solution allows any number of parties to be paid using the same tokens. A pricing contract is used to allow dynamic tariffs for each leg of the call route and to make tokens entity specific. The scheme uses prepaid hash chains to allow efficient verification of payment. In the mobile communications environment, the need for user authentication and online contact with a distant home location have been removed. This allows a mobile user to roam into any network and pay for services, with no need for roaming agreements or online contact with a third party. The need for interconnect agreements and trust between service providers has been eliminated.

We see the ability to pay any party, whether it is a large wide-area mobile operator or a home user providing a local Bluetooth connection, as a key enabler for ubiquitous mobile communications. Our scheme is an immense improvement over billing because it removes unnecessary trust between all parties and hence the need for roaming agreements, removes online communications with a home location, eliminates fraud due to tampering and falsification of records, provides fair dynamic charging, and allows real-time payment in any network by anyone.

The multi-party micropayment demonstrator validated the feasibility of using electronic payments to pay for mobile services. It was shown how the protocol could be integrated to pay for traffic from a wide variety of voice and data applications. The mobile wireless link was provided using both Bluetooth and WaveLAN, two technologies we envision being used to provide local area picocells. A JavaCard was used for chain storage but the prototype card used was found to be prohibitively slow. Performance measurements showed the computational efficiency of ongoing payments. The system bottleneck was the communications overhead of Java RMI, which, in its current form, is not suitable for handling multiple simultaneous ongoing payments.

The original multi-party protocol was extended and modified to produce a number of important contributions, most of which have not been previously considered in the research. A method for aggregating payments in core networks was designed, improving scalability by reducing state. A method was proposed to allow user payments to continue uninterrupted during handover between independent picocells. A novel consideration was how to make ongoing payments in ad-hoc networks. In order for public ad-hoc networks to succeed, the ability to make and receive payments will be required. The usability of the scheme was enhanced by allowing user-to-user payments and local area payments. Our scheme can now be used as part of a mobile wallet, paying not only for all communications as a user roams, but also paying for physical items from vending machines, or for passing money to other users.

In summary, a review of billing techniques exposed open problems to which we have responded with a multi-party micropayment protocol. The solution allows roaming in any network with offline guaranteed payment for all service providers without the need for trust agreements. The mobile user can pay any party in any location, with no ability to overspend.

Real-time payment liberates the mobile user by removing the control of the home network over which foreign networks can be used while roaming. The home operator no longer dictates the prices paid for using other mobile networks. Payment will allow independent local area wireless networks to flourish by providing faster, less expensive access to nearby mobiles. Such high-speed picocells will grow in number to cover entire cities. Any party will be able to offer value added services to any mobile located in any network, and receive payment. Mobiles will be able to pick and chose access networks and services wherever they roam. Ultimately, the scheme allows money to be used as easily and freely in a global network environment as cash is used today in the local marketplace. In conclusion, multi-party micropayments have the potential to revolutionise the telecommunications industry and transform the infrastructure of mobile communications.

8.2 Directions for Future Research

A number of avenues are possible to continue the research. This thesis represents a first attempt at a multi-party micropayment protocol aimed at mobile communications. It is likely that there are other ways of constructing a multi-party payment scheme, perhaps based on some of the other micropayment techniques examined in Chapter 3. There may also be other similar approaches which address some of the shortcomings of telecommunications billing.

The security analysis performed in Chapter 5 was informal and it would be prudent to perform a formal mathematical proof of the payment protocol before deployment. Likewise, the extensions presented in Chapter 7 were not implemented and it would be useful to validate and test them. As when we implemented the original multi-party protocol, it is likely that new issues will emerge and the proposed protocols will need to be modified accordingly when prototyped. With regard to the Java prototype, it would be useful to see how the protocol would perform when interfaced directly with the network layer without the overhead of RMI. Indeed, we have not addressed how the payment protocol would be integrated with the signalling protocols of existing mobile networks or the Internet. Another question is whether resources should be reserved as the pricing contract is assembled and how this could be integrated with existing reservation protocols. It may also be useful to consider how variable network conditions and congestion would impact on the payment system and charging mechanisms used.

Group payment, with two or more payees, could be investigated. For example, the call cost might be split between both the sender and receiver, or between any number of parties for a group call. The cost of a call to a mobile GSM user is split between both parties with the mobile user paying for the roaming leg between her current location and her home network. The bulk of the work with group payment will be deciding how to fairly divide the charges and how to deal with parties leaving or joining the group. While the problem of group security has been considered [WCSW+99], group payment seems to be a harder problem as it must later be proved to a broker who is entitled to payment. Tokens should only be redeemable by entities that were in the group at the time of payment, and not before or after the payment was made. In essence, non-repudiable proof of group state may need to be embedded within the payment.

The pricing contract allows dynamic tariffs to be set. However, we did not focus on suitable pricing or charging mechanisms. Ideally in our scheme, prices should remain fixed for the duration of a typical call, although charges can be reflected in a new pricing contract. Suitable adaptive pricing algorithms could be used to ease congestion across multiple service providers and economic theory could be applied here.

Different SPs will offer different tariffs, perhaps based on the time of day, current network load, or other factors. There needs to be a mechanism for the user to discover the least expensive route for her needs. If multiple access networks are in range there should be a mechanism to compare prices. Least cost routing to a mobile reachable through several access networks is also an issue. The pricing contract acts as a pricing

query and quotation protocol, but there may be more efficient ways to distribute pricing information in bulk. The same argument applies to QoS discovery, such as the need to locate the route that offers the best QoS within a specific price range. The local SP might provide price summaries or make routing decisions based on user stated preferences. Brokers will have knowledge of SP prices from redeemed contracts and it may be possible to make use of this information.

Optimal access network selection will also depend on the physical speed of the user and the likely duration of a call. The mobile device could be mobile-aware and intelligently pick different access networks based on a combination of mobility, call history profile, prices and QoS requirements. Smart algorithms to select or nominate appropriate enforcers based on call history would also be useful.

Chapter 1 introduced the concept of selling unused bandwidth on corporate mobile networks to passers-by. There will need to be a mechanism to discriminate between traffic from authorised employees and paying visitors. Network operators may also wish to recognise loyal customers and provide them with preferential tariffs. Use of specially identified payment chains may be one possible starting point. Selling unused bandwidth also raises the issue of how to handle current calls when that bandwidth is suddenly required again by the network operators. There needs to be a way to present such policies to users before payment begins. Another policy decision is how to deal with dropped calls.

Finally, we mentioned how the multi-party micropayment scheme can be used for payment, not only in mobile communications but also with wireline communications. It would be interesting to consider if the scheme would be useful in any other setting or application, in addition to payment.

Appendix A Cryptographic Benchmark Method and Measurements

In this appendix we explain how the cryptographic speed measurements were taken, to enable the results to be reproduced. We then present a summary of the measured data from which the charts and graphs in Chapter 4 were derived.

A.1 Benchmark Technique

A Java program was written to perform a number of iterations of each of the cryptographic algorithms and measure the time that each took. For the hash functions and symmetric ciphers we performed 50,000 iterations of each algorithm on a random 50KB message (51,200 bytes), and measured the total time for each in milliseconds. From this the number of kilobytes of input data that were processed per second (KB/s) was calculated by dividing the total data size (50000 * 50KB) by the time taken. We then calculated the number of 64 byte blocks processed at this speed per second. The results were used as the number of operations performed per second. We chose a 64-byte block as representative of a typical small message length.

For the RSA signature generation we used a 64-byte message, which was hashed using SHA1 to produce a 160-bit string, padded to give an encoded message of 127 bytes according to PKCS#1 [RSA99b], and encrypted with a 1024-bit RSA private key. It has been shown that proper padding before signing can defeat several possible RSA attacks [Bon99]. To verify the RSA signature required an RSA decryption with the public key, an SHA1 hash of the original message and a comparison of the two encoded hash values. The size of the message to sign has a negligible impact on the overall signature speed since the hashing operation is four orders of magnitude faster than the RSA encryption. Signing a 64-byte message took the same time as signing a 1-byte message and a 1KB message. Due to this, it is not useful to show the number of KB/s for RSA, and the data in Table A.2 shows the time taken for one operation instead. The number of operations per second for RSA refers to the number of signatures or verifications that can be performed per second. Direct RSA encryption of a full message rather than the hash is much slower than just signing the hash for messages greater than 1 KB. In addition, small public exponents were used in the RSA public key which allows exponentiation to be much faster, and the private key is stored in Chinese Remainder Theorem form which speeds up signing operations [SV93, RSA99b].

J/Crypto [Bal98a], the cryptographic library used for our measurements, obfuscates or hides sensitive data held in memory. This is done by XORing the sensitive data with a random string, and storing the result and the string after deleting the unprotected sensitive data. By default obfuscation is performed and this reduced the overall performance of the cipher algorithms by up to a factor of nine. We turned off J/Crypto obfuscation before taking timings, so as to be able to compare performance with other implementations which do not use it. PKCS#15 padding is always performed on DES and 3DES by J/Crypto and it was not possible to turn it off. The effects of this on small messages is discussed in Section 4.2.3. The padding used with hash algorithms is discussed in the same section. We used three-key Triple DES for our measurements as it has been shown to be more secure than two-key Triple DES [OW91].

A Sun implementation of the MD5 and SHA hash algorithms is included with Java 1.1. A standard Java security API is used to access these and the J/Crypto implementation. To avoid using the Sun versions by mistake we removed the Sun algorithms using the *Security.removeProvider("SUN")* method.

The timing measurements were made using the *System.currentTimeMillis()* method which returns the current time in milliseconds. The start time was recorded, the iterations were performed in a loop, and the end time

was recorded. The total time for all the iterations was measured instead of timing each one individually and summing to obtain a total. The reason for this is that some of the hash algorithms are so fast that they take less than a millisecond to execute and thus return a time of zero. By measuring a large number of iterations this problem was avoided. The time taken to perform the looping instructions themselves, as part of a non-empty loop, was found to be negligible, requiring only a few milliseconds and therefore not impacting on the results.

We used three different Java 1.1 JVMs from independent companies during our experiments. These were:

- Sun JDK 1.1.8 with Symantec JIT
- IBM JDK 1.1.8 with IBM JIT 3.5
- Microsoft VM 3188 with Microsoft JIT

Where not otherwise specified the Sun JVM was used by default.

To generate more accurate measurements we performed the benchmarking over long periods that ranged from minutes to several hours for each algorithm. This was necessary because we found that short runs produced significant timing differences depending on where the algorithm code was executed in the overall program. The effect was most noticeable with the IBM JVM where a code segment at the start of a program ran up to 50% slower than the exact same code segment appearing again later in the program. This effect was not present when the JIT was removed, and is perhaps due to the initial overhead of compiling the code for the first time. Over long timing periods the effect is negligible. For our measurements we used a single instance of an algorithm within a central loop in the program, as it might appear in a commercial application.

A.2 JVM Benchmarks

Table A.1 shows the average of three measurement runs using the JVM benchmark programs jBYTEMark [GR98] and VolanoMark [Nef99]. JBYTEMark performs calculations such as finding Fourier coefficients, Huffman compression, and use of linear equations, using primitive data types. VolanoMark is a Java server benchmark, which uses local loop back connections for group multicasts between a large number of threads, and is based on a chat server implementation. The higher the score in each benchmark, the better the performance.

jBYTEMark			
	SUN	IBM	Microsoft
Numeric sort	0.87702	0.96944	0.51974
String sort	1.98546	2.005516	1.24636
Bitfield	1.02249	0.8162	0.76018
FP Emulation	0.78655	2.52353	1.33098
Fourier	1.81757	2.10322	0.55601
Assignment	1.45898	2.05911	1.06512
IDEA	0.68956	0.83133	0.80536
Huffman	1.42031	1.88145	1.18162
Neural Net	1.85544	3.20027	1.61665
LU Decomp	0.38491	0.56685	0.25201
Integer Index	1.10417	1.44093	0.94325
FP Index	1.09085	1.5629	0.60959
VolanoMark 2.1.2			
Throughput	1134	2951	1017

Table A.1 JVM Benchmark Results on Three JVMs using jBYTEMark and VolanoMark

A.3 Cryptographic Algorithms on three different JVMs

The average of three runs of J/CRYPTO and RC6 algorithms on three different JVMs is shown in Table A.2. It shows the speed in KB/s and the corresponding number of operations possible per second on 64-byte messages.

	Sun			Microsoft			IBM	
	KB/s *	number/s		KB/s *	number/s		KB/s *	number/s
MD5	7205.27	115284		3074.951	49199		6062.14	96994
SHA-1	3349.402	53590		2087.926	33407		3647.221	58356
RC6	3315.333	53045		4466.667	71467		6100	97600
DES	1453.823	23261		547.6064	8762		1546.125	24738
3DES	643.0005	10288		291.5954	4666		610.6337	9770
RSA verify	0.00455	220		0.001967	508		0.004932	203
RSA sign	0.055572	18		0.020013	50		0.070798	14

Table A.2 Speed of Cryptographic Algorithms on Three Different JVMs

* The RSA measurements show the time, in seconds, per operation as calculated from 5,000 iterations, instead of KB/s as discussed earlier.

A.4 Cryptographic Algorithms with Small Message Sizes

The time taken to perform 500,000 rounds of each algorithm on messages with sizes from 1 to 264 bytes was measured. From this the number of operations per second on that specific message size was calculated, and the average obtained over three runs is shown in Table A.3.

Number per second					
Bytes	MD5	SHA-1	RC6	DES	3DES
0	93284	52400	30622	35542	14406
1	93284	52400	30622	35542	14406
8	92217	52460	30506	30534	12371
9	92575	52400	30486	29926	12251
16	92302	52400	30779	26201	10767
17	92217	52372	26265	25772	10694
24	93756	52285	26440	22912	9522
25	93650	52176	26402	22556	9463
32	93214	52372	26497	20292	8568
33	92575	52285	23340	19958	8496
40	93023	52659	23374	18366	7725
41	92047	52345	23358	18130	7684
48	94286	52433	23535	16763	7104
49	93475	52258	20833	16580	7064
55	94304	52372	20752	16623	7098
56	56110	27916	20694	15387	6568
57	56073	27843	20703	15214	6539
64	56338	28022	20825	14203	6096
65	55648	27891	18827	14076	6071
72	55426	27916	18721	13190	5690
73	55457	27916	18786	13076	5651
80	55396	27850	18886	12357	5297
81	55236	27883	17079	12259	5278
88	55748	27882	17100	11580	4988
89	55717	27868	17106	11496	4968
96	55586	27939	17192	10924	4697
97	55463	27891	15676	10847	4678
104	55810	27964	15691	10264	4450
105	55298	27931	15702	10198	4436
112	55816	27947	15676	9259	4120
113	55556	27835	13988	9193	4105
119	55910	27770	13974	9192	4117
120	39850	18976	13976	9295	4043
121	39717	18953	13954	9225	4032
128	40271	19022	14037	8827	3846
129	39670	18965	13172	8771	3836

Number per second					
Bytes	MD5	SHA-1	RC6	DES	3DES
136	39701	18968	13152	8433	3673
137	39670	18965	13145	8386	3656
144	39623	18969	13216	8094	3515
145	39604	18939	12427	8079	3504
152	39984	18957	12409	7767	3382
153	39850	18950	12413	7728	3374
160	39834	18991	12468	7464	3244
161	39686	18969	11756	7429	3238
168	39701	18983	11747	7173	3123
169	39670	18980	11745	7138	3112
176	39834	18976	11730	6878	3007
177	39768	18954	11185	6848	3008
183	39933	18976	11174	6854	3012
184	31037	14391	11164	6646	2914
185	30989	14378	11164	6615	2900
192	31260	14434	11206	6426	2809
193	30967	14397	10570	6396	2803
200	30958	14393	10557	6144	2714
201	30977	14397	10549	6121	2709
208	30889	14395	10589	5961	2630
209	30998	14388	10013	5933	2625
216	31209	14399	10002	5777	2553
217	31199	14388	9989	5755	2555
224	31137	14406	10027	5610	2485
225	31098	14399	9536	5609	2480
232	31096	14462	9534	5470	2406
233	31067	14458	9529	5453	2400
240	31158	14454	9549	5107	2279
241	31089	14443	9087	5090	2271
247	31122	14406	9001	5324	2328
248	25553	11629	9069	5215	2280
249	25504	11591	9072	5198	2277
256	25654	11621	9099	5066	2217
257	25498	11604	8702	5049	2211
264	25484	11600	8695	4919	2157

Table A.3 Speed of Cryptographic Algorithms with Small Message Inputs

A.5 Different Devices and Implementation Languages

	Number/sec					
Platform	PC	PC	PC	JavaCard	HP 620LX	PalmPilot
Language	Assembler	C++	Java	Java	C	C
MD5	1186944	789325	115284	44333	9419	797
SHA-1	477897	356338	53590	20014	4379	360
RC6	448430	276593	53045	15535	4334	279
DES	147059	101320	23261	6250	1901	204
3DES	53879	33752	10288	3378	841	68
RSA verify	2071.5	1381	220	50	17.97	0.95
RSA sign	55.5	37	18	4	1.47	0.22

Table A.4 Cryptographic Speeds Per Second on Various Platforms and Implementations

The speed of cryptographic algorithms on different devices and implemented using various programming languages is compared in Table A.4. This data was based in part on the following sources. The number of cycles per block of the assembly implementation of the hash functions and DES based ciphers was obtained from Preneel's performance data [Pre98] for a Pentium CPU. We used Aoki and Lipmaa's [AL00b] fast implementation of RC6 requiring 223 cycles per 128-bit block on a Pentium II, instead of the original submission requiring 254 cycles [RRSY98]. Knowing that our 400MHz Pentium II workstation could perform 400,000,000 cycles per second we were able to calculate the number of operations per second possible on a 64 byte input message. The RSA assembler speeds were scaled estimates based on our C++ speed measurements, which showed that assembler speeds were approximately 1.5 times faster than C++.

We used the Crypto++ [Dai99] software to obtain measurements on the 400MHz PC for all algorithms in C++. The Java measurements were based on the J/Crypto and RC6 measurements on the Sun JVM, as shown in Table A.2.

The JavaCard data is based on the specifications of the Infineon SLE 66CX320P 16-bit chip card IC [Inf99]. This chip has 64Kbyte ROM, 3Kbytes RAM, 32 Kbytes EEPROM, a 1100-bit Advanced Crypto engine for public key operations and a 64-bit DES accelerator. At the time of writing this is considered a high performance smart card. The IBM JavaCard, described in [BBEH+99, BBE99], used a similar but earlier version of the SLE 66X chip. Gemplus markets a commercial version of the IBM JavaCard under the GemXpresso [Gem98] brand. The JavaCard numbers for MD5, SHA and RC6 are scaled from the PC Java measurements, based on the relative performance of the algorithms to each other on the PC.

The handheld PC used was a Hewlett Packard HP620LX. We were unable to obtain cryptographic libraries for its Windows CE platform. Instead we wrote a program to perform standard byte operations such as addition, rotation and masking as would typically be found in a cipher implementation. We timed the execution of this program on the handheld and on the workstation and used the difference to obtain a scaling factor. The algorithm data was scaled based on this, providing an approximation of how actual algorithms would perform. An interesting discovery was that our program consistently executed 3% faster when a serial line was not attached to the handheld PC, as time was not spent by the operating system polling for data to arrive. We used the faster speed, although the difference does not significantly alter the results.

The PDA data is based on measurements of a C implementation on the PalmPilot IIIx with 4MB of RAM, taken by Daswani and Boneh [DB99]. Measurements were only available for SHA, DES and 512-bit RSA. Using the relative algorithm speeds observed with Crypto++ we estimated the speeds of the other algorithms.

A.6 Public Key Algorithm Speeds

Table A.5 shows public key speeds for signature generation and verification using Crypto++. The key sizes used were 1024-bit RSA, 1024-bit DSA, and 168-bit ECC using the GF(p) Nyberg-Rueppel signature scheme.

	Number of operations/second		
	RSA	DSA	GF(p)
Signature generation	37	170	123
Signature verification	1381	107	71

Table A.5 Signature Generation and Verification Speeds for Public Key Algorithms

Appendix B Micropayment Performance Calculations

The methods used to calculate performance data from high-level micropayment descriptions are summarised here. The basic assumptions made to allow different schemes to be directly compared and the choice of cryptographic algorithms is explained. Based on these design decisions, calculations can be made to predict performance of an implementation. For each of twenty selected micropayment schemes, predicted performance data is presented for storage space, communications bandwidth, and computational usage. A detailed example showing how these figures were calculated from one of the micropayment proposals is given. Other factors such as the overhead of maintaining and accessing a double spending database, and the type of digital signature used, are also discussed.

B.1 Scheme Selection

At least one representative scheme was chosen from each of our micropayment categories in Chapter 3. Where a scheme made a performance improvement over the fundamental method used by schemes in that category it was also included. In the case of hash chains we differentiated between credit and prepaid chains in order to highlight the performance differences. In addition our own improvements, such as infinite chains, combined UOBT-PayTree and PayFairer, were included to verify their contributions.

B.2 Choice of Cryptographic Algorithms

The choice of cryptographic algorithms will affect computational speed, as highlighted in Section 4.2. Storage space and message lengths will depend on key sizes, hash sizes, signature and certificate sizes, all of which depend in turn on the algorithms used. Table B.1 shows the algorithms selected. SHA performs slower than MD5 but has been shown to be more secure [Dob96]. RC6 was selected as one of the faster finalist candidate algorithms [Dra00] for the AES standard, and was shown to perform significantly faster than DES in software in Chapter 4. Recently the Rijndael algorithm, which has a similar performance to RC6 in Java, but with a slower key setup time [Dra00], has become the proposed AES standard, subject to final approval. RSA, with a 1024-bit key, was selected as the most suitable public key algorithm for fast signature verification, as discussed in Section 4.2.5.

Function	Algorithm	Key size (bytes)
Hashing	SHA1	-
Symmetric Ciphering	RC6	16
Public key Signatures	RSA	128

Table B.1 Cryptographic Algorithms used for Performance Evaluation

B.3 Storage

Based on the choice of cryptographic algorithms we can predict the size of specific constructions as given in Table B.2. An average message field, such as an identity or a URL, is taken to be 16 bytes. The exact size in an implementation will depend on the maximum possible values for that field and the encoding used, and is not specified by any of the schemes examined. In a global scenario sometimes the field size will be greater and other times it will be smaller. From our implementation in Chapter 6 we find this size to be a reasonable working value. In a hash chain scheme the user needs to record the position, or index, of the next unspent hash. Using a hash chain index of two bytes will allow a maximum chain length of 65,535. An average chain length of 1,000 is assumed.

Object	Size (bytes)
Average field	16
SHA hash value	20
Hash chain index	2
RC6 key	16
RC6 encrypted message	$\lfloor MsgSize/16 \rfloor * 16$
RSA private key (d)	128
RSA signature	128
RSA public key (e,n)	134
RSA public key certificate	310

Table B.2 Size of Basic Micropayment Constructions

RC6 uses 16 byte blocks and it is assumed that a message is rounded up to the nearest 16 byte multiple after encryption. An RSA digital signature with appendix is used as explained in Section B.7. The PKCS#1 standard for RSA signing specifies that a hash of the message, with an ASN.1 identifier for the message digest prepended, is padded and then encrypted with the private key to form the signature. The signed hash will be 128 bytes but the original message is also required to verify the signature. A small public exponent, e, of 2 bytes, with a 128 byte modulus n, and 4 bytes of length information as specified by ISO9797 [ISO91], is assumed for an RSA public key. When the public key (PK_U) is signed, along with the identity of the user (U), the broker (B), and an expiry date, each of 16 bytes, the total signed certificate will be 310 bytes:

$Cert_U = \{U, B, Expiry, PK_U\}, \{H(U, B, Expiry, PK_U)\}SK_B$

SK_B is the secret private key of the broker, and is used to generate the digital signature. A certificate owner will need to store their private key and the public key certificate. The certificate issuer, in this case the broker, keeps a copy of the certificate for retrieval and revocation purposes. A signature verifier, such as a vendor, will need to obtain the certificate to verify the signature but does not need to keep a copy of it once verified. For a user signed payment to a new vendor the user certificate is also sent with the signature for a new vendor. The broker is the certificate authority (CA) and a copy of the broker's public key, in a broker certificate, is needed to verify the signature on a user certificate. It is assumed that all parties that need the broker certificate have a copy of it before payment begins.

Table B.3 shows the storage space in bytes used by each party in the chosen schemes. Certificate and key space is separated from payment instrument material. As explained in Section 4.3.1.1, the user is assumed to hold €10.00 and to spend it with a single vendor in 1,000 payments, showing the storage costs before payment at the user, after payment at the vendor, and after redemption at the broker.

In several schemes, such as SubScrip and prepaid hash chains, a micropayment instrument must be bought for each new vendor. Where this is not specified by the scheme we include the overhead of performing a secure transaction with the broker. Space limitations prevent all the implementation decisions for each protocol being explained, but the important choices are summarised.

Content delivery and receipt stages are ignored in all schemes, as they are not part of actual payment. Temporary memory space required during a transaction is not included. Mini-Pay has three extra fields, related to the daily spending limit, in the user-spending certificate. A prepaid hash chain commitment requires an additional field, over a credit commitment, to specify the total value of the prepaid chain. It is assumed that the broker certificate is required to securely buy a prepaid chain, but is not essential in a credit scheme. Only the extended part of an infinite chain is considered, to be able to compare the performance of extending a chain rather than generating a new one. A binary PayTree with a depth of 5 is assumed. The

	Storage space (in bytes)								
	Certificates/keys			Payment Instrument Material			Total		
	User	Vendor	Broker	User	Vendor	Broker	User	Vendor	Broker
Tang	16	16	32	0	0	80000	16	16	80032
NetBill	598	828	1058	0	0	212000	598	828	213058
Mini-Pay	470	1090	1090	0	240000	240000	470	241090	241090
H.chain (credit)	438	310	748	22	200	180	460	510	928
H.chain (prepaid)	310	310	438	218	216	216	528	526	654
Infinite chain	0	0	0	22	720	700	22	720	700
PayTree	310	310	438	852	288	788	1162	598	1226
UOBT	438	310	748	24	832	188	462	1142	936
UOBT PayTree	438	310	748	232	288	788	670	598	1536
MicroMint	0	0	0	80000	80000	20000	80000	80000	20000
MicroMint Ext.	0	0	0	200000	100000	40	200000	100000	40
SubScrip	438	620	1058	16	32	192	454	652	1250
Millicent	40	20	40	232	145	40.125	272	165	80.125
PayFair	16	16	32	22	42	68	38	58	100
PayFairer	16	16	32	22	60	20	38	76	52
SVP	16	16	32	68	116	16	84	132	48
Wheeler's Bets	438	310	748	22	200	180	460	510	928
Lottery Tickets	438	310	748	22	242	200	460	552	948
Yen	438	310	748	22	242	200	460	552	948
Jarecki	470	310	780	18	224	264	488	534	1044

Table B.3 Micropayment Storage Space Requirements

UOBT is assumed to consist of 32 chains derived from a single chain of length 32, requiring 1024 hashes to generate the whole structure. UOBT-PayTree has 32 chains and a depth of 5. The MicroMint user holds and spends 1,000 coins. With MicroMint extensions a minimum range of 10 hashes is required to prevent forgery, but only 50% of a range of coins is sent.

In SubScrip certificates are assumed necessary for the initial vendor macropayment. Millicent customer secrets are used to authenticate users and are included under key costs, while scrip secrets are part of the payment material. Millicent scrip consists of 7 fields of which one is a hash value. Millicent secrets are used to generate MACs, and are therefore set to be 20 bytes, the size of an SHA1 hash. The first and most secure version of the SVP protocol is used. To improve storage in the Jarecki scheme the registration message is set to contain a hash of the user certificate, instead of the full certificate. The vendor then does not have to store the full certificate, a storage saving of over 300 bytes.

B.4 Communication

When examining micropayment communication we are interested in the number of independent connections and the size of messages sent over those channels. Only communication that takes place during payment is used, as offline-clearing messages will not impact on the transaction speed. Table B.4 shows the volume of messages passing between parties during payment and the total number of messages sent. Where there are multiple message exchanges between two parties, such as with Wheeler's bets and lottery tickets, the total traffic size that travelled either way across the link is shown. If there is a user-broker or vendor-broker message during payment then this is an online broker interaction.

The micropayment schemes assume a reliable transport layer and time will be taken to initially setup a network connection. For example, TCP [Pos81, Com00] uses a 3-way message handshake to establish a connection and the additional latency due to the message round trip time is significant. This extra delay will be present only for the first payment in a session, as the connection can stay established if payment is ongoing. The more network connections that need to be established, as in online schemes, the greater the delay overhead.

	Communication bandwidth (in bytes)							
	First Payment				Average Payment			
	U-V	U-B	V-B	#msgs	U-V	U-B	V-B	#msgs
Tang	208	0	464	3	208	0	464	3
NetBill	784	0	880	3	448	0	880	3
Mini-Pay	582	0	0	1	582	0	0	1
H.chain (credit)	510	0	0	1	20	0	0	1
H.chain (prepaid)	216	324	0	3	20	0	0	1
Infinite chain	700	0	0	1	20	0	0	1
PayTree	288	0	0	1	20	0	0	1
UOBT	1118	0	0	1	20	0	0	1
UOBT PayTree	598	0	0	1	20	0	0	1
MicroMint	80	0	0	1	80	0	0	1
MicroMint Ext.	103	0	0	3	103	0	0	3
SubScrip	210	0	320	4	34	0	0	2
Millicent	272	408	0	4	272	0	0	2
PayFair	80	0	196	3	20	0	0	1
PayFairer	60	134	0	3	20	0	0	1
SVP	132	0	0	3	132	0	0	3
Wheeler's Bets	65.1	0	0	3	60.2	0	0	3
Lottery Tickets	550	0	0	2	20	0	0	1
Yen	550	0	0	2	20	0	0	1
Jarecki	546	0	380	3	20	0	0.04166	1.0025

Table B.4 Micropayment Communications Usage

Assumptions made for each protocol are now outlined. Mini-Pay will be online with a broker if the credit-spending limit is exceeded, but normally this will not occur. With a credit hash chain the user certificate is sent to the vendor in the first payment. No certificate needs to be sent for a prepaid chain. A secure communication with the broker is necessary to buy a prepaid chain. With our infinite chain the first payment includes the one-time signature of the extended chain. With PayTree and UOBT-PayTree the multiple chains are present and the broker is not contacted online for a new vendor chain. Just one coin is sent in a 1-cent MicroMint payment. The MicroMint extensions vendor challenge is for three more coins in the range, and each can be expressed as one byte relative to the start of the range.

With Rivest's lottery tickets the initial user request for the vendor chain final hash is assumed to be part of the pre-purchase protocol and is not included. The probability of an actual payment occurring in Wheeler's coin flips, Rivest's lottery tickets, and Yen's improvement is set to be 0.01. Thus, a payment takes place every 1 in 100 times and is worth €1.00. The probability value could be made smaller but then the payment amount would be greater and a macropayment might have to be used. In Jarecki the average number of polling messages by a user spending up to a €20.00 credit limit is 5. This yields a 0.25% chance of a poll message for every 1-cent payment. The probability of an unnecessary user warning being triggered by a user who spends up to their credit limit is 0.01, as obtained from the figures in [JO97].

B.5 Computation

Figure 4-4 shows that the speed of hashing and symmetric encryption depends on the size of the input message. RC6 ciphering becomes progressively slower on every 16 byte input increment. Figure 4-5 showed that the amount of computation that can be performed will depend on the hardware architecture. This is likely to be a mobile device or commodity computer for the user, while cryptographic accelerators and clustered computers may be used by active merchants and brokers. Therefore relative computation speeds are used rather than exact ones for a specific platform.

	Input Bytes											
	0-16	17-32	33-48	49-55	56-64	65-80	81-96	97-112	113-119	120-128	129-144	145-160
SHA	1	1	1	1	1.89	1.89	1.89	1.89	1.89	2.75	2.75	2.75
RC6	1.7	1.98	2.23	2.51	2.51	2.77	3.05	3.34	3.73	3.73	3.96	4.2

Table B.5 Relative Cryptographic Speeds for Varying Input Sizes

Using the data in Table A.3 the speeds of SHA hashing and RC6 ciphering for different sized inputs is calculated relative to the speed of an SHA hash on input less than 56 bytes. The resulting relative speeds are shown in Table B.5, and allow computation speeds to be calculated given a specific input message. The figures in Table A.2, for the Sun JVM, are used to obtain a relative computation speed for RSA signature generation and verification. The input size does not significantly impact on RSA speed, as discussed in Appendix A, and so a constant figure is used for all input sizes. In terms of the time it takes to perform 1 SHA hash, it takes 238 times longer to verify an RSA signature, and 2910 times longer to generate one.

Computation (1 unit = 1 SHA hash)						
	First Payment			Average Payment		
	User	Vendor	Broker	User	Vendor	Broker
Tang	3.75	2.51	12.97	3.75	2.51	12.97
NetBill	6085	6332	3416	2924	3181	3416
Mini-Pay	2910	476	0	2910	238	0
H.chain (credit)	3910	477	0	49.5	1	0
H.chain (prepaid)	1000	239	2910	49.5	1	0
Infinite chain	1770	225	0	49.5	1	0
PayTree	1117	243	0	49.5	1	0
UOBT	3965	477	0	31	1	0
UOBT PayTree	1149	481	0	49.5	1	0
MicroMint	0	4	0	0	4	0
MicroMint Ext.	0	5	0	0	5	0
SubScrip	2910	238	3148	0	0	0
Millicent	7.48	9.28	13.87	5.5	9.28	0
PayFair	1001	2	1004.79	49.5	1	0
PayFairer	1001.89	3.51	6.63	49.5	1	0
SVP	3.75	5.75	0	3.75	5.75	0
Wheeler's Bets	39.1	4.77	0	0.495	1.01	0
Lottery Tickets	3910	1477	0	49.5	45.5	0
Yen	3910	527	0	49.5	1.01	0
Jarecki	3910	239	3386	49.5	1	0

Table B.6 Micropayment Computational Requirements

Table B.6 shows the computational requirements for the first payment and subsequent payments. Computation is only shown for parties active in the actual payment, and does not include offline-clearing performance. Assumptions made about computation for the protocols are now outlined.

With NetBill the merchant does not need to check the user signature, as the broker will perform this online. Also the user does need to obtain a Kerberos ticket for each new vendor encountered. A time-space trade-off is used with hash chains in that every 100^{th} hash of a chain of length 1000 is cached, reducing the computation by a factor of 10 to obtain the next hash to release. Hash chain calculation is included, although this can be performed offline. An electronic cheque scheme requiring a user signature is used for the SubScrip macropayment. In Millicent, when new vendor scrip is purchased the new vendor customer secret is returned encrypted with the broker customer secret. Millicent request and response MACs require a large input to be hashed. SVP assumes the use of a secure MAC based on a one-way hash function. We use HMAC [KBC97, BCK96] as a secure means of keying a hash function [BCK96]. Simply using a hash (H) of the

message (M), concatenated with the key (K), H(M,K), is not as secure [KR95, Sch96]. HMAC performs two hash functions, and the key *K* must be 20 bytes, the length of the hash output:

HMAC = H(K ? opad, H(K ? ipad, M))

where opad and ipad are two different fixed strings, the same length as the hash output.

B.6 Example Performance Calculations

PayTree is used as an example to illustrate how the performance figures were calculated. PayTree is described in Section 3.3.3.5, and the performance figures are summarised in Table B.7. It is a prepaid instrument where the broker signs the root of the tree. Therefore only a normal broker certificate, of size 310 bytes, is required by all parties. The broker will hold the corresponding private RSA key of size 128 bytes, giving a total of 438 bytes at the broker.

	User	Vendor	Broker	#msgs
Certificate storage (bytes)	310	310	438	-
Payment inst. (bytes)	852	288	788	-
Communications, 1st pay.	288	0	0	1
Communications, average pay.	20	0	0	1
Computation, 1st pay	1117	243	0	-
Computation, average pay.	49.5	1	0	-

Table B.7 Performance of PayTree

A binary PayTree with depth of 5 is assumed. The user must store sufficient information to regenerate the tree. A hash chain is attached to each of the $2^5 = 32$ leaves, and the root hash of each chain, must be recorded. Since a hash is 20 bytes this requires (32 * 20) = 640 bytes. The signature on the base of the PayTree is 128 bytes, and using message appendix, as described in Section B.7, the signed message, in this case the tree root hash of 20 bytes, is also stored. Each PayTree chain is spent at a different vendor and users can return to vendors with a partially spent chain. Recording the current 2 byte position in each of the 32 chains requires (32 * 2) = 64 bytes. The payment instrument total for the user is therefore (640 + 128 + 20 + 64) = 852 bytes.

A vendor must store the last received hash, 20 bytes, the tree signature (128 + 20), and the tree authentication path. A tree of depth *m* requires *m+1* nodes to authenticate a leaf as part of the signed tree. Therefore (6 * 20) = 120 bytes are needed. Vendor storage is (20 + 128 + 20 + 120) = 288 bytes. Different parts of the tree will be redeemed by different vendors. To prevent double redeeming the broker stores the root of each redeemed chain (32 * 20) and the signed tree root (128 + 20), yielding broker storage of 788.

For a first payment to a vendor the user sends the first payment hash from a new chain (20), the authentication path from that chain's leaf to the tree root (6 * 20), and the signed tree root (128 + 20), giving a total of 288 bytes. The vendor does not need to reply and the broker is offline. For a subsequent payment to the same vendor only the next hash in the chain (20) is sent. The index position does not need to be sent, as this will be the number of hashes performed to obtain the previous payment hash. Again, only a single message is needed.

Each hash chain is assumed to be of length 1000. For the first payment the user must re-compute the chain, taking 1000 hashes, and derive the nodes necessary for the tree authentication path. For a tree of depth *m*, the calculation of the entire tree from the leaves requires 2^m-1 hashes. Hashing two nodes, $h(x_1, x_2)$, where each is 20 bytes, requires the same computation as a single hash due to the padding of SHA. The verification of an

authentication path from the key nodes requires *m* hashes. Therefore the calculation of the key nodes from the leaves requires (Calculation of tree – Calculation from key nodes/authentication path) = $2^m - 1 - m$. A user signature is assumed when the tree is bought from a broker, but this occurs only once for every 32 chains. Therefore the total user computation is ((2910/32) + 1000 + 2^5 – 1 – 5) = 1117. For the first payment the vendor must verify the tree signature (238) and the tree authentication path (5), giving a total of 243.

For subsequent payments the vendor only needs 1 hash to verify the payment token. The user re-computes the chain from the last cached point. With 10 cached points in a chain of length 1000 the average amount of hashing is 49.5 hashes.

B.7 Digital Signature with Appendix or Message Recovery

The type of digital signature used in a micropayment protocol will affect performance in terms of the computation, communications bandwidth and storage required. In Section 4.2.5 we showed that signatures based on the RSA algorithm were the most computationally efficient for micropayment schemes, due to the fast signature verification. However, independent of the underlying cryptographic algorithm, signatures may be implemented in two different ways, each now described in turn.

A *digital signature with appendix* requires the original plaintext message in order to be able to verify the signature. A hash of the message is signed and this signature is sent along with the original message:

M, $\{H(M)\}SK_X$

If the message is tampered with its hash will not match the signed hash, giving an invalid signature. The PKCS#1[RSA99b] standard defines a digital signature scheme with appendix using the RSA algorithm.

A *digital signature with message recovery* allows the entire message to be obtained from the signature material when the signature is verified, without need for a copy of the original message:

$\{M\}SK_X$

ISO/IEC 9796 [ISO91] specifies an RSA message recovery scheme. With RSA the signature is generated by encrypting the material to sign, *M*, with the user's private key, *d*, to form ciphertext *C*:

$C = M^d \bmod n$

The signature is verified by decrypting the ciphertext with the user's public key, *e*:

$M = C^e \bmod n$

Intuitively, the smaller the size of the message *M* the faster the computation should perform. However all messages are padded with redundancy up to the modulus size before exponentiation. This is to prevent forgery attacks as mentioned below. With a 1024-bit modulus (key size) the signature size will also be 1024 bits. However with an appendix scheme the original message must also be sent. Therefore a message recovery scheme will minimise the micropayment bandwidth and storage costs provided that the message is small enough to fit in the signature. For ISO 9796, using a *2k*-bit modulus the message must be equal or smaller than *k* bits. For calculations in our micropayment analysis it would be efficient to use RSA appendix signature for messages greater than 512 bits, and an RSA message recovery signature for messages below this size.

Unfortunately a number of attacks [CNJ99, CHJ99, Gri00] against RSA message recovery signatures, in particular ISO9796, were published in 1999. These attacks have resulted in the recommended withdrawal of ISO 9796 [ISO99a]. Since there is no longer a secure RSA message recovery standard, we will use only RSA with appendix signatures, as defined in PKCS#1, in our analysis. This results in a maximum extra transmission of 512 bits per signature, depending on the message length.

The ISO 9796 attacks were against the redundancy or padding function used in the message recovery signature, rather than the RSA algorithm. A redundancy function is necessary to prevent two independent signatures, *s1* and *s2*, being joined to form a single valid signature *s*, using multiplication:

$s1 = m1^d, \quad s2 = m2^d, \quad \Rightarrow \quad s = s1.s2 = m1^d.m2^d = (m1.m2)^d \mod n$

where *m1* and *m2* are two independent messages, and *d* is the private signing key. A redundancy function *R*, which is not multiplicative, can prevent this.

$R(m1.m2) \neq R(m1).R(m2)$
$s1 = R(m1)^d, \; s2 = R(m2)^d$
$\Rightarrow s = (R(m1).R(m2))^d$ which is not a valid signature on m1.m2

Use of a secure hash function as part of the redundancy function, as in PKCS#1, prevents these attacks.

B.8 Micropayment Double Spending Database

An additional overhead will be present if databases, not held in memory, need to be accessed during payment. In each micropayment scheme the vendor will typically have to perform a DB lookup on the first payment to ensure that the payment instrument is fresh and is not a replay. For example with a hash chain he must check that this is a new chain, with Millicent that the scrip has not already been spent, and with MicroMint that this coin has not already been accepted. Two exceptions to this are Wheeler's bets and SVP. With bets it is not in the interest of either party to use an old value as it allows the other party to fix the outcome. With SVP the vendor gives the user a fresh random nonce to include in each new payment, which is an alternative method to prevent replay at the cost of an extra network message.

Alternatively the double spending DB size can be limited by forcing the user to include a time dependent nonce in the payment instrument. Only very recent payments then need to be kept in the DB for checking. The slowest part of a DB lookup is the time taken to access the data on the hard disk. This delay is due to the disk seek time, rotational latency, and the data transfer. Hard disks have typical access times of 5 to 10 ms. Therefore approximately 100 single table DB lookups can be performed per second on a commodity computer, assuming a total data access and processing time of 10ms.

The size of the double spending DB at a vendor will depend on the number of payments received. To minimise the hard disk access overhead a large memory should be used to hold as much of the DB as possible.

Most of the schemes do not require additional DB lookups after the first payment as the relevant information can be cached in memory. The exceptions to this are the schemes which use the exact same payment instrument format for every payment, including the first payment. Such schemes are MicroMint and Millicent. However, where small ranges of consecutive serial numbers are used, as in Millicent, each serial number may be represented by a single bit and can be held in memory. This is not possible with MicroMint

due to the randomness of the coin values. While Millicent may only require 1 bit per payment to prevent double spending, the hash chain schemes only require one hash (20 bytes) per chain spent. If more than 160 payments are made on average per chain, then the hash chain schemes are as efficient in terms of the DB size.

B.9 Macropayment Performance Estimates

The five sample macropayment systems for which performance is estimated are SSL [FKK96], WTLS [WAP99a], SET [MV97], Mandate II E-cheque [Man98], and fully anonymous Ecash [Sch98]. The macropayment performance estimates are based on the referenced published protocols. Table B.8 shows the communications usage in bytes for the user, vendor, and broker in each macropayment system. Data is shown for the first contact with a new vendor and for further payments to that same vendor. The computational cost, expressed as a number of SHA1 hashes, for each party during the first payment and ongoing payments is shown in Table B.9. A number of assumptions were made when estimating the performance of each protocol, and these are now outlined.

Payment Sys.	First Payment (Bytes)				Average Payment (Bytes)			
	U-V	U-B	V-B	#msgs	U-V	U-B	V-B	#msgs
SSL	1317	0	0	6	152	0	0	2
WTLS	1435	0	0	6	152	0	0	2
SET	3143	0	3212	6	3143	0	3212	6
E-cheque	924	0	0	2	924	0	0	2
Ecash	840	0	1036	3	840	0	1036	3
H.chain (credit)	510	0	0	1	20	0	0	1
MicroMint	80	0	0	1	80	0	0	1
Millicent	272	408	0	4	272	0	0	2

Table B.8 Communications Usage for selected Macropayment and Micropayment Systems

Most SSL users do not possess digital certificates, and so only SSL server authentication takes place, and this was the case used. Both MD5 and SHA were used with 1024-bit RSA, and 16 byte symmetric keys. We used RC6 as the symmetric cipher, even though this is not currently available in the standard SSL cipher suites. The reason for this was that our micropayment estimates had used RC6. We used an SSL certificate size of 791 bytes, based on an example X.509 certificate given in [Bal98].

SSL does not specify payment messages and so we used a payment message consisting of the amount (6 bytes), a credit card number (16 bytes), an expiry date (4 bytes), the cardholder's name (20 bytes) and address (50 bytes), giving a total payment message size of 96 bytes. A 16 byte acknowledgement message was assumed to be returned in response to a verified payment. However typically an e-commerce site will return an entire Web page, which would be several hundred bytes, in response to a user purchase. We did not consider this as part of payment. The same payment message sizes were used with WTLS. It was assumed that ongoing payments take place in the same SSL session, or in a re-established session, despite some server clusters being unable to resume SSL sessions [APPS00]. Multiple SSL handshake messages were assumed to be placed into a single SSL record message, before being transmitted across the network. It must be noted that SSL record structures and field sizes have been optimised in the protocol specification, unlike our micropayment estimates.

WTLS was not designed to provide end-to-end security between a WAP user and an Internet merchant. Instead WTLS provides security from the mobile user to the WAP gateway server, usually located at the mobile network operator. At the WAP server WTLS to SSL translation occurs with a secure SSL connection established between the WAP server and the merchant's Web server. The disadvantages of this are that no

Payment Sys.	First Payment (#SHA1 hashes)			Average Payment (#SHA1 hashes)		
	User	Vendor	Broker	User	Vendor	Broker
SSL	545	2965	online	14.03	14.03	online
WTLS	3474	3474	online	13.46	13.46	online
SET	4345	12851	16224	4345	12851	16224
E-cheque	3152	3152	0	3152	3152	0
Ecash	310	2910	4476	310	2910	4476
H.chain (credit)	3910	477	0	49.5	1	0
MicroMint	0	4	0	0	4	0
Millicent	7.48	9.28	13.87	5.5	9.28	0

Table B.9 Computational Requirements for selected Macropayment and Micropayment Systems

client authentication can be provided to the merchant, even if a user WTLS certificate is used, and the computational overhead, and resulting latency, will be doubled due to having two secure connections rather than one. Large merchants and institutions have adopted two approaches to circumvent these problems and remove the extra SSL connection. The first solution is to locate the merchant server at the mobile operator's WAP server, making the service specific to that mobile network. Another approach is to use a WAP server with a WTLS protocol stack at the merchant server on the Internet, and have the operator gateway transparently forward the WTLS traffic over the Internet to this merchant server. In our WTLS estimates we assume that one of these approaches is used, and that no additional SSL connection is present.

We use mutual authentication with WTLS, requiring a WTLS user certificate. Such a certificate is likely to be issued when a user purchases a WAP device, or signs up with a mobile operator. WTLS certificates are optimised for size. An example optimised certificate, given in [WAP00], is 425 bytes and we use this. There is an abbreviated WTLS handshake protocol that assumes the use of shared secret keys. Since a user will not have keys established when they first deal with a new merchant we do not use this. There is also an optimised handshake protocol which assumes that the merchant obtains the user certificate from an online public directory. However, since the certificate is only 425 bytes, we use the normal handshake protocol where it is sent over the wireless interface to the merchant for the first payment. RSA based key exchange is used, and reported as the only available method during initial handshake. HMAC is used for MAC generation and key calculation, and requires two SHA hashes as with SSL MAC generation.

A SET certificate size of 693 bytes was used, based on the example cardholder certificate given with the protocol specification. The user has only one public key pair, used for signing, but the vendor and payment gateway both have two, with the second used for encryption. Capture of a previously authorised payment usually takes place offline in batches at the end of the day, and was therefore not included in our calculations.

With E-cheque each smart card holds two certificates with the corresponding private keys. We assumed a certificate size of 310 bytes, the same as used in the micropayment estimates, as no example was provided. One public key pair is used for generating signatures while the other is used for encryption.

The Ecash implementation uses RSA and Triple DES. We used a 1024-bit RSA keys and used the 3DES rate from Table A.3 in Appendix A. We used 3DES rather than RC6, because Ecash does not provide cipher suites like SSL or WTLS. We assumed an average Ecash payment to consist of five coins, which is likely because no change can be issued if anonymity is to be maintained.

Appendix C Multi-Party Micropayment Performance Calculations

C.1 Performance Assumptions and Estimates

The calculations used to estimate performance of both the original and optimised multi-party micropayment protocol are given in this section. We base the estimates on a call involving three SPs, of which the first in the call route is the enforcer. RSA keys of 1024 bits and the SHA hash function are used. Unless otherwise stated we use the figures in Table B.2 as the size of the basic micropayment constructions. These are the same figures used in our analysis in Chapter 4.

Object	Size (bytes)
SP/Broker Identity	4
Chain length	2
Expiry (month)	1
SP TID	4
SP charge	4
Value (monetary)	2
Call_request	16

Table C.1 Size of Basic Protocol Fields

Specific field sizes are given in Table C.1 and from these the size of constructs consisting of several fields are derived in Table C.2. Since users are not identified in the protocol a 4-byte entity identifier is adequate allowing up to 2^{32} possible SPs and brokers. Chain length will not exceed 2^{16} and can be represented using 2 bytes. The 1-byte expiry field indicates the month a chain expires, as suggested in Section 5.4. Each SPs local part of the TID is 4 bytes, giving a combined TID of 12 bytes, with 3 SPs.

Object	Contents	Size of contents	Total bytes
Signed $Comm_P$	P_0, Len, Value, Enforcer, Expiry, Sig_{Broker}	20+2+2+4+1+128	157
Unsigned $Comm_E$	E_0	20	20
Unsigned contract	TID, SPs, Charge, $Comm_P$, P_{Start}, Start, P_value, $Comm_E$, R_brokers	12+12+12+157+20 2+2+20+12	249
Fully signed contract	Unsigned contract, 3 SP signatures	249+3(128)	633
Optimised signed contract	Unsigned contract, Enforcer signature	249+128	377

Table C.2 Size of Multi-Party Micropayment Components

Table C.3 summarises the computation, storage and communications costs of the protocol. The operations used and the components sent and stored are listed for the user, enforcer and ordinary SP. The computational cost for hashing and signatures are based on the figures in Section B.5. The computational costs are based on the user verifying all 3 SP signatures on the contract. The enforcer must verify 2 SP signatures, the broker signature on $Comm_P$, generate his own signature, and perform a hash to verify the user request. The other SPs verify 2 SP signatures, 1 broker signature, and add their own signature to the contract. A chain holder must perform *(n-1)/2* hashes on average to obtain the next hash to release, where *n* is the length of the chain, or the length between cached hash values if parts of the chain are held in memory. For both user and enforcer we assume every 100 hashes are cached.

Entity	Contents	Contents size (bytes)		Total	Optmised
Storage (bytes)					
Certificate storage					
User	$Cert_{Broker}$, $Cert_{Enf}$	310+310		620	620
Enforcer	$Cert_{Broker}$, $Cert_{Enf}$, SK_{Enf}	310+310+128		748	748
SP	$Cert_{Broker}$, $Cert_{Enf}$, $Cert_{SP}$, SK_{SP}	310+310+310+128		1058	438
Payment material					
User	$Comm_P$, P_N, Index_X	157+20+2		179	179
Enforcer	$Comm_P$, Amt, Contract, P_X, X, E_Y, Y	157+4+633+20+2+20+2		838	445
SP	Contract, P_X, X, E_Y, Y	633+20+2+20+2		677	421
Computation (#hashes)					
First payment					
User	Verify $Sig_{Enf, SP2, SP3}$	238*3		714	238
Enforcer	Verify $Sig_{Enf, SP2, SP3}$, $Comm_P$, P_X, Sign contract	(238*3)+1+2910		3625	3149
SP	Verify $Sig_{Enf, SPX}$, $Comm_P$, Sign contract	(238*3)+2910		3624	238
Average payment					
User	(n-1)/2 hashes to get P_X	(100-1)/2		49.5	49.5
Enforcer	Verify P_X, (m-1)/2 hashes to get E_Y	1+(100-1)/2		50.5	5.5
SP	Verify P_X, E_Y	1+1		2	2
Communications (bytes)					
First payment			#msgs		
User <-> Enf	$Comm_P$, P_{X-1}, X-1, Call_req, Signed contract, P_X, $Cert_{SP2}$, $Cert_{SP3}$	157+20+2+16+476+ 20+310+310	2	1311	435
Enf <-> SP	Unsigned contract (SP1), Partial contract (SP2, SP3), Full contract, $Cert_{SP3}$, P_X, E_Y	217+(249+2(128))+ 633+310+20+20	4	1705	843
SP <-> SP	Unsigned contract (SP1, SP2), Partial contract (SP3), Full contract, $Cert_{Enf}$, P_X, E_Y	233+(249+128)+ 633+310+20+20	4	1593	1169
Average payment					
User	P_X	20	1	20	20
Enforcer	P_X, E_Y	20+20	1	40	40
SP	P_X, E_Y	20+20	1	40	40

Table C.3 Performance Calculations for Multi-Party Micropayment Protocol

During contract signing three messages pass between each pair of SPs. While the message size remains constant in step three, it grows in size as each SP adds entries or signs in steps one and two. Each contract signature is a private key encryption of a hash of the contract, requiring 128 bytes. A fully signed contract with 3 SP signatures is therefore 633 bytes. SP certificates must also be distributed in order to verify the pricing contract. The user will already hold both broker and enforcer certificates. Each SP should already hold the certificate of any neighbouring SP to whom he is directly connected. Therefore the user needs to be provided with certificates of SP2 and SP3, while the enforcer needs the certificate of SP3, and SP3 needs the enforcer certificate. Transfer of these certificates form part of the communications messages for the first payment, as shown. Each entity maintains an index position of the last spent hash value, counted as part of the storage.

C.2 Optimisations

Optimisations derived in Section 5.10 are applied to the basic protocol to obtain improved performance estimates, listed in Table C.3. The optimised figures were obtained by removing all non-enforcer SP signatures, by removing non-enforcer SP verification of the payment commitment, by not sending the commitment back to the user, and by reducing enforcer endorsement hashing using closer chain cache points. Removing SP signatures not only reduces computation but also results in a smaller signed contract, as shown in Table C.2, and hence reduced storage and communications.

Appendix D Prototype Implementation and Measurement Data

Technical details concerning the prototype implementation are now presented. We examine the structure, contents, and purpose of the messages which are passed between entities across the network, and between threads of execution within a single entity. A summary view of the behaviour of the main user, SP, and broker processes are then given in state diagrams. Each process diagram shows when specific messages can be received, the actions taken due to each message received in a particular state, and any messages that are sent in response. These diagrams capture the interactions and behaviour of the payment system at a high level. We then describe the payment objects which are used to hold and manipulate payment tokens, hash chains, and pricing contracts. These form part of the messages and are also used for managing payment information within each entity. Finally a summary of computational and communications related measurement data, obtained from experimenting with the prototype, is presented.

D.1 Message Types

In Chapter 6 we explained how a single Message class was designed to allow data to be passed across the network, and between communicating threads on the same node. Each RMI call sends the Message class as the only parameter. The attributes and methods of this generic Message object are shown in Figure D-1. The contents attribute holds one of the many possible message types, and can either be another object or simply a string of characters. For example the contents might be a payment message object or a redeem request object. In addition to this contents object there are attributes specifying the message type, the identity of the entity that sent the message, and the identity of the intended receiver. The list of possible message types, with each assigned a byte identifier, is also shown.

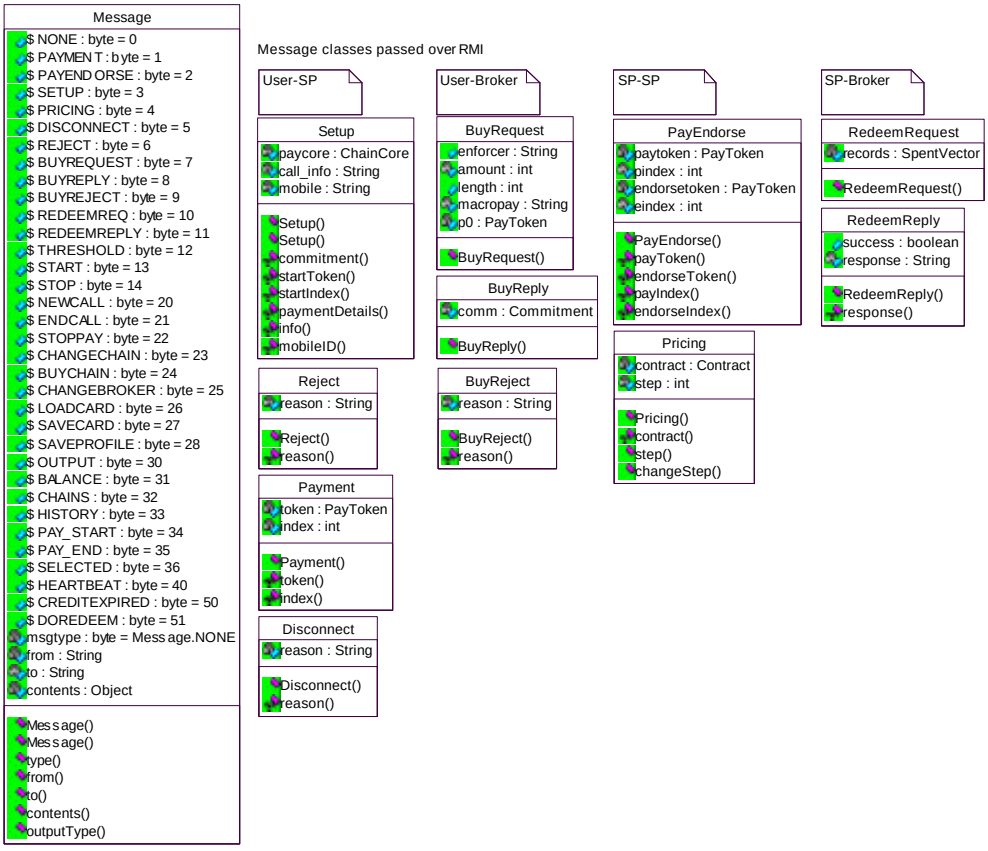


Figure D-1 Class Diagram of Payment Message Classes sent over RMI

Figure D-1 also shows the attributes and operations of each of the payment message classes. We use the Unified Modeling Language (UML) [BRJ99] to model and visualise our different message classes in the class diagram. For each class the class name, attributes, and methods are shown respectively. For example, the Setup class contains the call setup information and payment commitment from the mobile user. The PayEndorse class carries both payment and endorsement tokens. The class diagrams in this section were produced using Rational Rose for Java, and edited to show all attributes for each class. The payment objects that are present in the payment messages, such as PayToken and ChainCore, are discussed in Section D.3.

Message	Purpose	Contents
RMI Network Msgs:		
Payment	User Payment Hash	PayToken, index
PayEndorse	Enforcer-endorsed payment	Paytoken, EndorseToken, indexes
Setup	User call setup request	Commitment, Start token and index, call details, Mobile address
Pricing	Pricing contract	Contract in one of three stages (fill, sign, verify)
Disconnect	Terminate current active call	Reason (such as user hangup or invalid token)
Reject	Reject current call request	Reason (such as invalid chain or token)
BuyRequest	Buy payment chain from broker	Encrypted anchor, length, enforcer, amount, macropayment
BuyReply	New broker-signed commitment	Signed payment commitment
BuyReject	Refuse a user BuyRequest	Reason (such as invalid macropayment)
RedeemReq	Redeem spent contracts	Sequence of pricing contracts, and highest pay/endorse tokens
RedeemReply	Indicate success/failure of redeem	Redeem details including failed contracts
Threshold	Set how many SPs share a relay	Number of Start messages required to start relay
Start	Start traffic forwarding at relay	SP identity
Stop	Stop traffic forwarding at relay	SP identity
Mobile GUI msgs:		
Newcall	Call destination for new call request	Destination address
Endcall	User pressed terminate call	
Stoppay	User pressed stop payment	
ChangeChain	Change currently selected chain	Chain number in chains vector
BuyChain	Buy a new payment commitment	Enforcer, amount, length
ChangeBroker	Change default broker	Broker RMI identity
LoadCard	Load chains from smart card	Smart card PIN
SaveCard	Save chains to smart card	Smart card PIN
SaveProfile	Save user profile on smart card	Smart card PIN, user profile (call history, preferences)
Output	Output message to GUI	String message
Balance	Update balance counter window	User held-value in cents
Chains	Update chain graph window	Vector of payment chains
History	Update call history window	Vector of all spent chains and pricing contracts
Pay_Start	Replace Pay button with Stop button	
Pay_End	Replace Stop button with Pay button	
Selected	Update current selected chain	Chain number in chains vector
Relay/Monitor:		
Heartbeat	Time to release another payment	#Tokens to release
CreditExpired	User credit for last token has run out	Reason for expiry (time elapsed or volume sent)
DoRedeem	Redeem spent contracts	Broker identity
AddCredit	Extend user credit at relay	Time/Volume indicator, extra milliseconds/bytes allowed
ResetCredit	Reset credit counters to zero	

Table D.1 Purpose and Contents of each Different Message Type in the Payment System

Other classes exist for passing information between threads, such as communication with the user GUI or a traffic monitor. The purpose and contents of each possible message type, that is passed as part of the generic Message object, are listed in Table D.1. For messages that carry a single simple data type, such as an integer or a string, an additional separate class for holding the message contents is not needed. The possible communications paths for each type of message are shown in Figure 6-4.

D.2 Entity Processes

Having described the types of messages, and their contents, that can be passed between entities we now describe the behaviour of each entity payment process. We present state diagrams for each entity, showing what messages they can receive, the actions they take based on receiving a message, and any messages that are sent in response.

We use the Specification and Description Language (SDL) [ITU99b, OFPR+94] to present the behaviour of each process as a finite state machine. SDL is a description language for communications systems, widely used in the design of telecommunications services. SDL defines symbols to represent a state, an input message received, an output message transmitted, a decision taken, an action performed, and a procedure executed, amongst others. Each entity remains in a specific state until the arrival of an input message, or signal, which can cause the process to move to another state, after some optional processing has been performed.

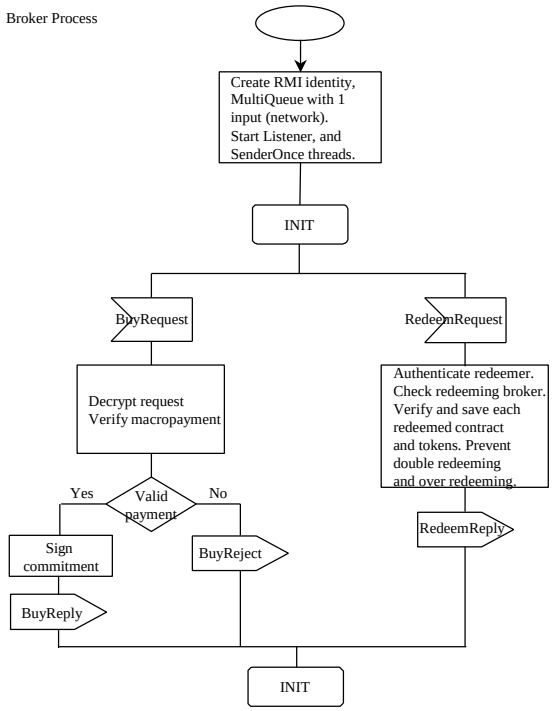


Figure D-2 Broker Payment Process

SDL process diagrams for the main payment process running in each entity are now presented. These diagrams show the behaviour of the broker, user and SP payment components. The SP process contains enforcer functionality, which is triggered if a payment commitment specifying the current SP as the enforcer arrives in a call setup message. The SDL diagrams do not show exactly to whom or from whom messages are received, although this can be ascertained from Figure 6-4 and our protocol description in Chapter 5. For example, the enforcer sends the PayEndorse message both upstream and downstream, ensuring that both the SPs and the mobile receive it, regardless of the position of the enforcer in the call route. Upstream refers to the direction from where the payment source originates, that is in the direction of the mobile user, while downstream is the opposite direction, that is the direction in which payment flows away from the mobile through the call SPs.

Figure D-2 shows the state diagram for the broker payment process, which can receive and process payment chain purchase requests and redeem requests. Figure D-3 and Figure D-4 show the behaviour of the user payment process. Some of the messages passed between the payment process and the user GUI are not shown, due to space constraints. Figure D-5, Figure D-6 and Figure D-7 show the SP payment process. Figure D-8 shows a procedure used by this process to decide which direction in the current call route to send a message. While the diagrams cannot capture all the details that take place in each process, they do provide a useful view of the messages that can be received or sent in each particular state.

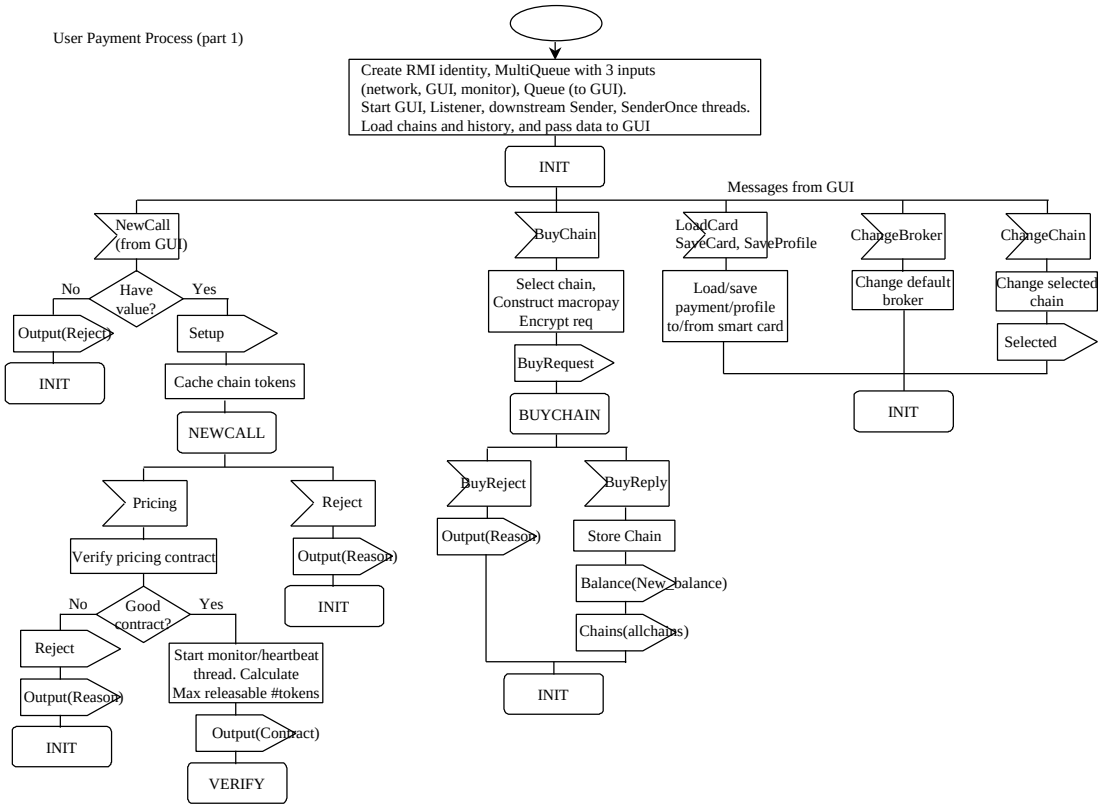


Figure D-3 User Payment Process – Part 1

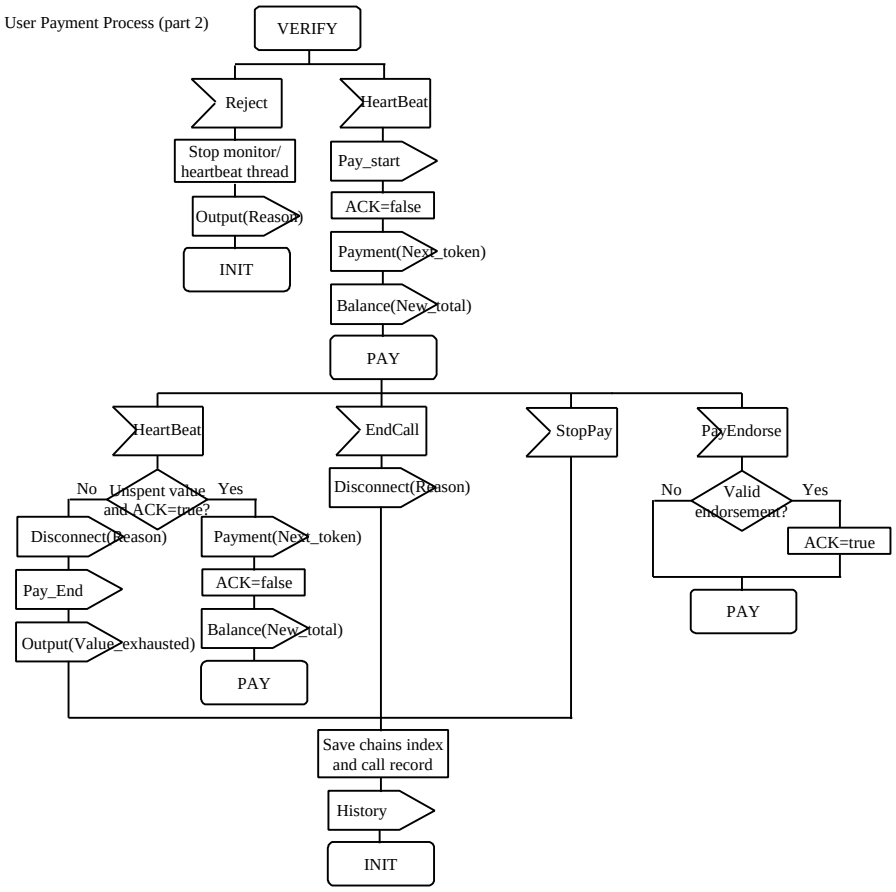


Figure D-4 User Payment Process – Part 2

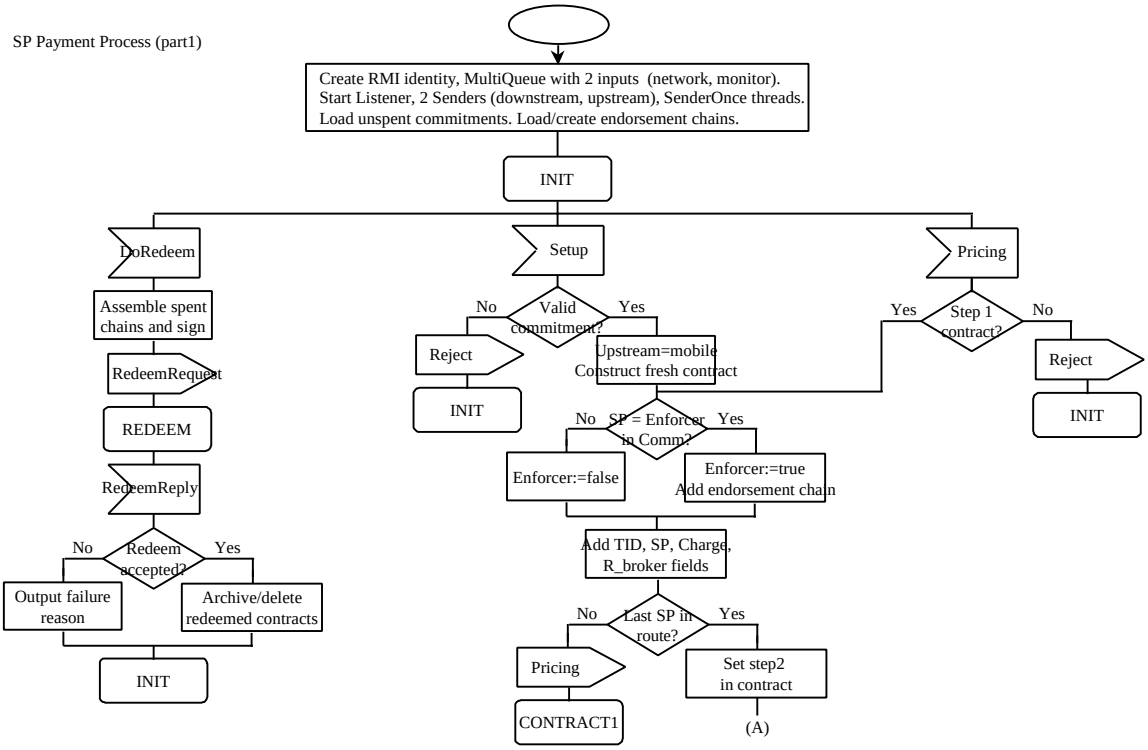


Figure D-5 SP Payment Process – Part 1

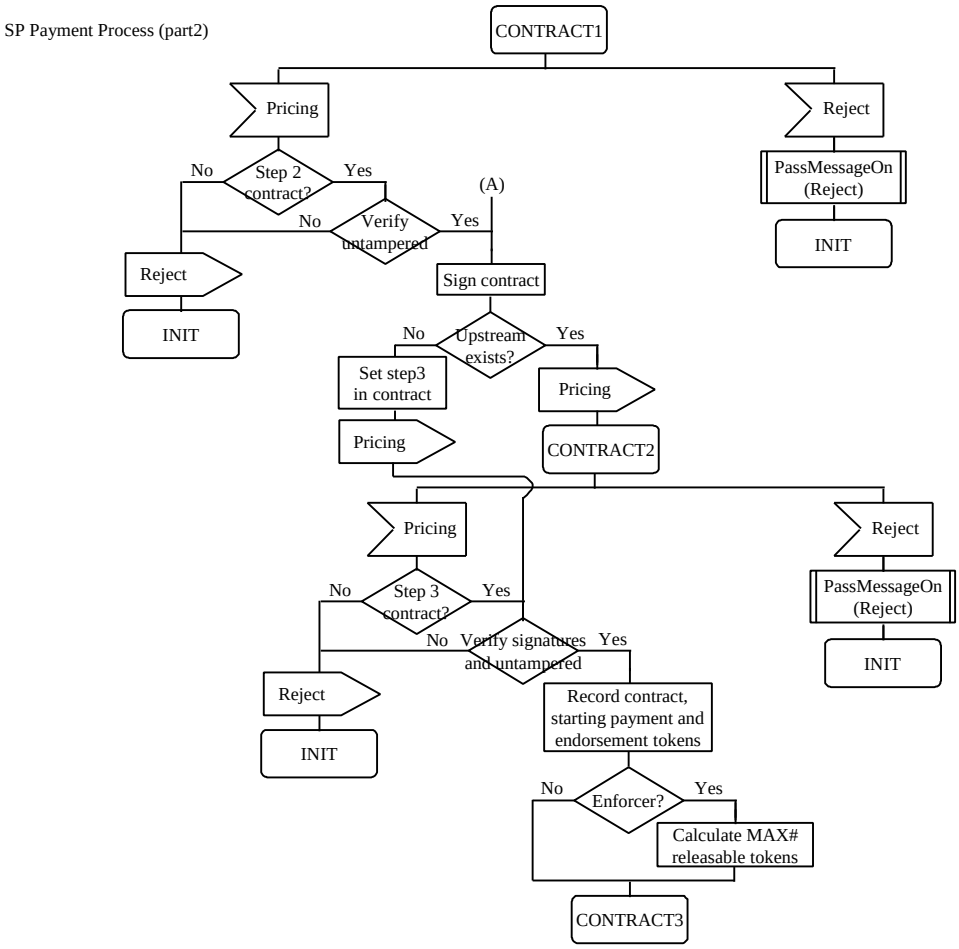


Figure D-6 SP Payment Process – Part 2

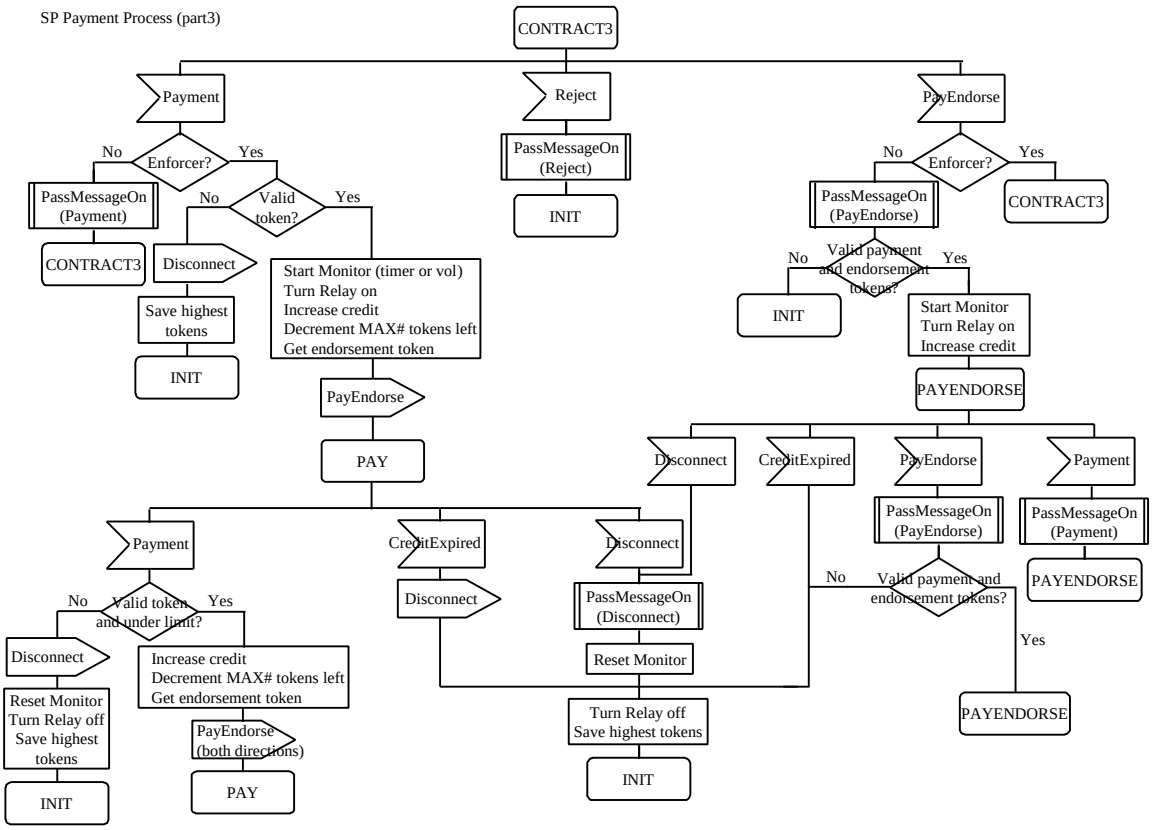


Figure D-7 SP Payment Process – Part 3

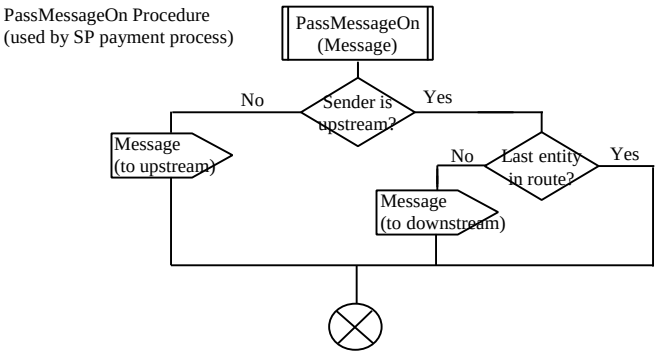


Figure D-8 PassMessageOn Procedure used by SP Payment Process

D.3 Payment Classes

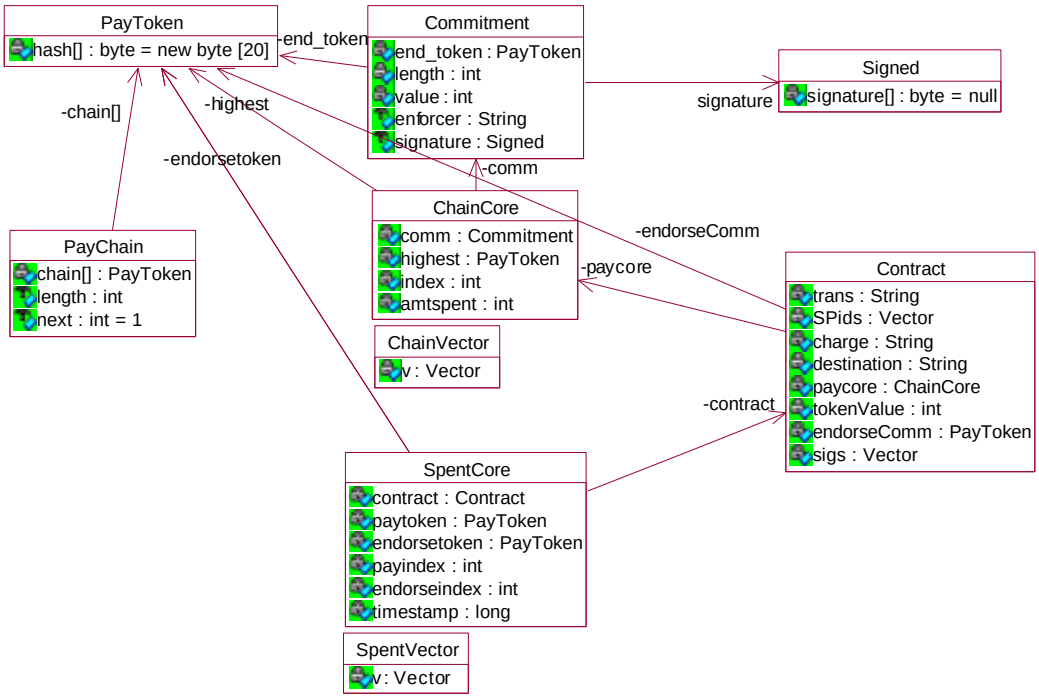


Figure D-9 Class Diagram of Payment Objects

Figure D-9 shows the attributes of the main payment classes used within the system. Due to space constraints we do not show the methods associated with each class. The PayToken class holds a single payment hash, which was defined to be a 20-byte SHA1 hash. Endorsement tokens are also 20-byte SHA1 hashes and use the same class. The PayChain class stores all or a subset of tokens from a payment chain for quick access from memory when releasing tokens. In contrast, the ChainCore object holds only the data necessary to re-generate a chain from the highest held token. The ChainVector holds a sequence of ChainCore objects. The SpentCore class contains all the payment information that needs to be stored after a call, for redemption by an SP, or for call history records at the user. In turn the SpentVector holds a sequence of SpentCore records. The signed class is used to hold an RSA signature produced by the J/Crypto signing function. The pricing contract is assembled and held in the Contract class, and contains a variable number of signatures depending on the call route.

D.4 Measurement Data

A summary of the measurement data collected during experimentation, and from which the charts of Chapter 6 were derived, is now given in table form. Table D.2 shows the number of computations that can be performed per second by the prototype on both a mobile laptop and fixed host. Table D.3 and Table D.4 show the time to perform specific payment transactions, such as setting up a newcall, and show the computational time as part of this overall time. Table D.5 shows the maximum number of transactions possible per second on the RMI prototype with a mobile Bluetooth user. Finally, Table D.6 shows the delays experienced when communicating with the prototype JavaCard.

Procedure	Number/s	
	233MHz	400MHz
User		
Verify contract	50	
Get next token	95329	
Enforcer		
New contract	27	43
Payment	15152	24570
Add endorsement	87951	266667
SP		
Step1 contract	200	500
Step2 contract	18	48
Step3 contract	53	111
PayEndorse	7299	11848
Broker		
Process BuyReq	18	50
Process RedeemReq (10c)	1.3	4.3
Newcall computation		
New_con+step2+step3	9	18.9
step1+step2+step3	12.7	31.3

Table D.2 Number of Computational Payment System Procedures Possible Per Second

Action	Time (ms)			
	Ethernet	WaveLAN	Bluetooth	Computation included in action:
User				
Buy	201	233	305	Chaingeneration(1000), process BuyReq
Newcall	937	1007	1088	New contract, 2(step1), 3(step2), step3, verify contract
Payment	54	76	108	Get next token, Payment, add endorsement, 2(PayEndorse)
Empty msg	35	46	67	
Enforcer				
Assemble contract	724			New contract, 2(step1), 3(step2), step3
SP				
Redeem (10c)	1673			Process RedeemReq

Table D.3 Total Time to Perform System Transactions

	Time (ms)				
Action	User_calc	Enforcer_calc	SP_calc	Broker_calc	Communications
Redeem (10c)	0	0	0	786	887
Empty msg	0	0	0	0	67
Payment	0.01049	0.07737	0.2214	0	107.69074
Assemble contract	0	111	83	0	530
Newcall	20	111	83	0	874
Buy	57	0	0	55	193

Table D.4 Computation and Communication Times for System Transactions

Procedure	Comms and computation ms	#/second	Computation only #/second
Enforcer			
New contract	46.205	22	43
Step 2&3	60.855	16	33
Payment	16.418	61	24570
SP			
Step 1	45.873	22	500
Step 2	74.89	13	48
Step 3	50.188	20	111
PayEndorse	23.591	42	11848
PayEndorse (no comp.)	23.679	42	
Newcall setup			
Newcall at enforcer	107.06	9	19
Newcall at SP	170.951	6	31

Table D.5 Throughput Measurements for System Transactions and Computation Procedures

	Time (ms)						
#Chains	1	2	4	8	12	16	20
# of chain bytes	221	440	876	1748	2620	3492	4364
Load chains	9719	14267	26734	54313	87469	119359	156344
Load chains and profile	11109	15735	28437	56110	89422	121406	158438
Save chains	8265	12704	22141	43282	67813	93531	122437
Load profile	3922						
Save profile	4890						
Connect and disconnect	2078						

Table D.6 Time to Load/Save Chains and User Profile with JavaCard

Bibliography

[3GPP99a] 3G TS 32.105: 3rd generation partnership project; technical specification group services and system aspects; 3G charging; call event data, version 0.0.1, December 1999.

[3GPP99b] 3GPP TR 23.922: 3rd generation partnership project; technical specification group services and systems aspects; architecture for an all IP network, version 1.0.0, October 1999.

[3GPP00] 3G TS 33.102: 3rd generation partnership project; technical specification group services and system aspects; 3G security; security architecture, version 3.5.0, July 2000.

[3GPT99] 3GPP2 P.S0001: Wireless IP network standard, version 1.0, December 1999.

[3GPT00] 3GPP2 SC.P000X: IP network architecture model for CDMA2000 spread spectrum systems, version 0.0.2, March 2000.

[AAH00] B. Aboba, J. Arkko, and D. Harrington. Introduction to accounting management. IETF Internet Draft, June 2000.

[Abo99] B. Aboba. Certificate-based roaming. IETF Internet Draft, April 1999.

[ACGH+00] B. Aboba, P. Calhoun, S. Glass, T. Hiller, P. McCann, H. Shiino, G. Zorn, G. Dommety, C. Perkins, B. Patil, D. Mitton, S. Manning, M. Beadles, P. Walsh, X. Chen, T. Ayaki, S. Sivalingham, A. Hameed, M. Munson, S. Jacobs, T. Seki, B. Lim, B. Hirschman, R. Hsu, H. Koo, M. Lipford, Y. Xu, E. Campbell, Shinichi Baba, and E. Jaques. Criteria for evaluating AAA protocols for network access. IETF Internet Draft, April 2000.

[ACPZ00] J. Arkko, P. Calhoun, P. Patel, and G. Zorn. DIAMETER accouting extension. IETF Internet Draft, June 2000.

[AFB98] W. Almesberger, T. Ferrari, and J. Le Boudec. SRP: a scalable resource reservation protocol for the Internet. In *Proceedings of the 6th International Workshop on Quality of Service (IWQoS'98)*, pp. 107-16, Napa, California, May 1998.

[AJSW97] N. Asokan, P. Janson, M. Steiner, and M. Waidner. State of the art in electronic payment systems. *IEEE Computer*, 30(9):28-35, September 1997.

[AK96] R. Anderson and M. Kuhn. Tamper resistance –a cautionary note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pp.1-11, Oakland, California, November 1996.

[AL00a] B. Aboba and D. Lidyard. The accounting data interchange format (ADIF). IETF Internet Draft, April 2000.

[AL00b] K. Aoki and H. Lipmaa. Fast implementations of AES candidates. In *Proceedings of the 3rd AES Candidate Conference*, New York, April 2000.

[AMS96] R. Anderson, H. Manifavas, and C. Sutherland. NetCard - a practical electronic cash system. In *Proceedings of the 4th Security Protocols International Workshop (Security Protocols)*, pp. 49-57, Lecture Notes in Computer Science vol. 1189. Springer-Verlag, Berlin, 1996.

[And97] R. Anderson. The formal verification of a payment system. Computer Laboratory, Cambridge University, UK, 1997.

[And98] M. Anderson. The electronic check architecture. Financial Services Technology Consortium (FSTC) White Paper, September 1998.

[APPS00] G. Apostolopoulos, V. Peris, P. Pradhan, and D. Saha. Securing electronic commerce: reducing the SSL overhead. *IEEE Network*, 14(4): 8-16, July 2000.

[Ark99] J. Arkko. Accouting requirements. IETF Internet Draft, October 1999.

[ASPT98] ACTS Project AC095 ASPECT. Project final report and results of trials. ASPECT Deliverable 20, December 1998.

[ATW96] N. Asokan, G. Tsudik, and M. Waidner. Server-supported signatures, In *Computer Security - ESORICS '96 Proceedings*, pp. 131-43, Lecture Notes in Computer Science vol. 1146. Springer-Verlag, Berlin, 1996.

[ATW97] N. Asokan, G. Tsudik, and M. Waidner. Server-supported signatures, In *Journal of Computer Security*, 5(1):91-108, 1997.

[Azb97] I. Azbel. PayWord micro-payment scheme: strengths, weaknesses and proposed improvements. Department of Computer Science, University of Cape Town, South Africa, 1997.

[Bal98a] Baltimore Technologies. J/Crypto guide for developers, version 3.0. IFSC House, Dublin, Ireland, 1998.

[Bal98b] H. Balakrishnan. Challenges to reliable data transport over heterogeneous wireless networks. Ph.D thesis. University of California at Berkeley, August 1998.

[BAN90] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18-36, February 1990.

[Ban00a] M. Banan. LEAP: one alternative to WAP. Free Protocols Foundation PD 108-102-02, Bellevue, Washington, March 2000.

[Ban00b] M. Banan. The WAP trap – an expose of the wireless application protocol. Free Protocols Foundation, Bellevue, Washington, May 2000.

[BB00] N. Brownlee and A. Blount. Accouting attributes and record formats. IETF Internet Draft, June 2000.

[BBC+94] J. Boly, A. Bosselaers, R. Cramer et al. The ESPRIT project CAFE - high security digital payment systems. In *Computer Security - ESORICS '94 Proceedings*, pp. 217-30, Lecture Notes in Computer Science vol. 875. Springer-Verlag, Berlin, 1994.

[BBCD+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Request for Comments: RFC 2475, December 1998.

[BBE99] M. Baentsch, P. Buhler and T. Eirich. Java embedded technology – Zurich (JetZ). IBM Zurich Research Laboratory Report RZ3169, Zurich, Switzerland, 1999.

[BBEH+99] M. Baentsch, P. Buhler, T. Eirich, F. Horing, and M. Oestreicher. JavaCard – from hype to reality. *IEEE Concurrency*, 7(4):36-43, October 1999.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Keyed hash functions and message authentication. In *Advances in Cryptology – CRYPTO '96 Proceedings*, pp. 1-15, Lecture Notes in Computer Science vol. 1109. Springer-Verlag, Berlin, 1996.

[BCS94] R. Braden, D. Clark and S. Shenker. Integrated services in the Internet architecture: an overview. Request for Comments: RFC 1633, June 1994.

[BDL97] D. Boneh, R. DeMillo, and R. Lipton. On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology – EUROCRYPT '97 Proceedings*, pp.37-51, Lecture Notes in Computer Science vol. 1233. Springer-Verlag, Berlin, 1997.

[Ber98] A. Berendt. Accounting for the future. *Telecommunications International*, 32(5):35-40, Horizon House Publications, May 1998.

[BFN96] T. Berners-Lee, R. Fielding, and H. Nielsen. Hypertext transfer protocol – HTTP/1.0. Request for Comments: RFC 1945, May 1996.

[BG92] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *Advances in Cryptology – CRYPTO '92 Proceedings*, pp. 390-420, Lecture Notes in Computer Science vol. 740. Springer-Verlag, Berlin, 1992.

[BG97] Banksys/Groupement Cartes Bancaires. Interoperable chip-secured electronic transactions (C-SET) protocol specification. Information Society Initiatives in Standardization (ISIS) Project, November 1997.

[BGH+95] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. iKP – a family of secure electronic payment protocols. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*. pp.89-106, New York, USA, July 1995.

[BGSB96] H. Beadle, R. Gonzalez, R. Safavi-Naini, and S. Bakhtiari. A review of Internet payments schemes. In *Proceedings of the Australian Telecommunication Networks and Applications Conference (ATNAC'96)*, pp.486-94, Melbourne, Australia, December 1996.

[Blu82] M. Blum. Coin flipping by telephone: a protocol for solving impossible problems. In *Advances in Cryptology: A Report on CRYPTO 81*, pp. 11-15, ECE Report 82-04, Dept. of Electrical and Computer Engineering, U. C. Santa Barbara, 1982.

[Blu99] Bluetooth Consortium. Specification of the Bluetooth system. Volumes 1 and 2, version 1.0, December 1999.

[BM94] D. Bleichenbacher and U. Maurer. Directed acyclic graphs, one-way functions and digital signatures. In *Advances in Cryptology – CRYTPTO '94 Proceedings*, pp. 75-82, Lecture Notes in Computer Science vol. 839. Springer-Verlag, Berlin, 1994.

[Bog00] K. Bogestam. Paying your way in the mobile world. *Telecommunications International*, 34(1):57-8, Horizon House Publications, January 2000.

[Bon99] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society*, 46(2):203-13, February 1999.

[Bra93] S. Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology – CRYPTO '93 Proceedings*, pp. 302-18, Lecture Notes in Computer Science vol. 773. Springer-Verlag, Berlin, 1993.

[Bra99] S. Brands. Rethinking public key infrastructures and digital certificates – building in privacy. Ph.D thesis, Eindhoven University of Technology, The Netherlands, September 1999.

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *The unified modeling language user guide*. Addison-Wesley, Reading, Massachusetts, 1999.

[BS97] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology – CRYPTO '97 Proceedings*, pp. 513-25, Lecture Notes in Computer Science vol. 1294. Springer-Verlag, Berlin, 1997.

[Bur98] J. Burstein. An implementation of MicroMint. M.Sc thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1998.

[BV98] L. Blunk and J. Vollbrecht. PPP extensible authentication protocol (EAP). Request for Comments: RFC 2284, March 1998.

[BZBH+97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) – version 1 functional specification. Request for Comments: RFC 2205, September 1997.

[CAN98] ACTS Project AC014 CANCAN. Final report. September 1998.

[CB97] D. Chaum and S. Brands. Minting electronic cash. *IEEE Spectrum 34(2):30-4*, February 1997.

[CBHL95] S. Crocker, B. Boesch, A. Hart, and J. Lum. CyberCash: payments systems for the Internet. In *Proceedings of the 5th Internet Society Conference on Internet Networking (INET'95)*, Honolulu, Hawaii, June 1995.

[CDFF+99] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A framework for multiprotocol label switching. Internet Draft, September 1999.

[Cer97] Certicom. Current public-key cryptographic systems. Certicom White Paper, Ontario, Canada, April 1997.

[CFMS97] T. Cramer, R. Friedman, T. Miller, D. Seberger, R. Wilson, and M. Wolczko. Compiling Java just in time, *IEEE Micro*, 17(3):36-43, May 1997.

[CFMY96] A. Chan, Y. Frankel, P. MacKenzie and Y. Tsiounis. Mis-representation of identities in e-cash schemes and how to prevent it. In *Advances in Cryptology – ASIACRYPT '96 Proceedings*, pp. 276-85, Lecture Notes in Computer Science vol. 1163. Springer-Verlag, Berlin, 1996.

[CFN88] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology – CRYPTO '88 Proceedings*, pp. 319-27, Lecture Notes in Computer Science vol. 403. Springer-Verlag, Berlin, 1988.

[CG97] D. Carrel and L. Grant. The TACACS+ protocol version 1.78. IETF Internet Draft, January 1997.

[CGKT+00] A. Campbell, J. Gomez, S. Kim, Z. Turanyi, C. Wan and A. Valko. Design, implementation and evaluation of cellular IP. *IEEE Personal Communications*, 7(4):42-9, August 2000.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84-8, 1981.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology – CRYPTO '82 Proceedings*, pp. 199-203. Plenum Press, New York, 1983.

[Cha85] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM 28(10):1030-44*, October 1985.

[Cha87] D. Chaum. Blinding for unanticipated signatures. In *Advances in Cryptology – EUROCRYPT '87 Proceedings*, pp. 227-33, Lecture Notes in Computer Science vol. 304. Springer-Verlag, Berlin, 1987.

[Cha88] D. Chaum. Blind unanticipated signature systems. U.S. Patent 4,759,064, U.S. Department of Commerce, July 1988.

[CHJ99] D. Coppersmith, S. Halevi, and C. Jutla. ISO 9796-1 and the new forgery strategy. IEEE P1363 Standard Specifications for Public Key Cryptography Research Submission. August 1999.

[CHTY96] J. Camp, M. Harkavy, J. Tygar, and B. Yee. Anonymous atomic transactions. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pp.123-33, Oakland, California, November 1996.

[CINT97] CIBERNET. Cellular industry billing exchange record (CIBER). Washington DC, 1997.

[CLR97] Y. Cho, Y. Lin, and H. Rao. Reducing the network cost of call delivery to GSM roamers, *IEEE Network*, 11(5):19-25, September 1997.

[CM99] S. Corson and J. Macker. Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations. Request for Comments: RFC 2501, January 1999.

[CMC99] S. Corson, J. Macker and G. Cirincione. Internet-based mobile ad hoc networking. *IEEE Internet Computing*, 3(4):63-70, July 1999.

[CMO99] P. Chaudhury, W. Mohr, and S. Onoe. The 3GPP proposal for IMT-2000. *IEEE Communications*, 37(12):72-81, December 1999.

[CMS96] J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In *Computer Security - ESORICS '96 Proceedings*, pp. 33-43, Lecture Notes in Computer Science vol. 1146. Springer-Verlag, Berlin, 1996.

[CNJ99] J. Coron, D. Naccache, and J. Stern. On the security of RSA padding. In *Advances in Cryptology – CRYPTO '99 Proceedings*, pp. 1-18, Lecture Notes in Computer Science vol. 1666. Springer-Verlag, Berlin, 1999.

[CNP98] J. Chomicki, S. Naqvi, and M. Pucci. Decentralized micropayment consolidation. In *Proceedings of the 18th International Conference on Distributed Computing Systems*, pp. 332-41, Amsterdam, The Netherlands, May 1998.

[CO95] D. Cunningham and D. O'Mahony. Secure pay-per-view testbed. In *Proceedings of the 2nd IEEE International Conference on Multimedia Computing and Systems (ICMCS'95)*, pp. 308-11, Washington D.C., May 1995.

[COBU96] ACTS Project AC031 COBUCO. Cordless business communication system (COBUCO) system concept and architecture. COBUCO Deliverable 01. February 1996.

[COBU98] ACTS Project AC031 COBUCO. Final report. COBUCO Deliverable 20, October 1998.

[Col99a] M. Collins. Telecommunications crime, part 1. *Computers & Security*, 18(7):577-86, 1999.

[Col99b] M. Collins. Telecommunications crime, part 2. *Computers & Security*, 18(8):683-92, 1999.

[Col00] M. Collins. Telecommunications crime, part 3. *Computers & Security*, 19(2):141-8, 2000.

[Com00] D. Comer. *Internetworking with TCP/IP: principles, protocols, and architectures*. 4th edition, Prentice Hall, New Jersey/London, 2000.

[CP92a] D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO '92 Proceedings*, pp. 89-105, Lecture Notes in Computer Science vol. 740. Springer-Verlag, Berlin, 1992.

[CP92b] D. Chaum and T. Pederson. Transferred cash grows in size. In *Advances in Cryptology – EUROCRYPT '92 Proceedings*, pp. 390-407, Lecture Notes in Computer Science vol. 658. Springer-Verlag, Berlin, 1992.

[CP93] R. Cramer and T. Pederson. Improved privacy in wallets with observers. In *Advances in Cryptology – EUROCRYPT '93 Proceedings*, pp. 329-343, Lecture Notes in Computer Science vol. 765. Springer-Verlag, Berlin, 1993.

[CPP99] K. Chen, R. Prasad, and H. Poor (Editors). Special issue on software radio. *IEEE Personal Communications*, 6(4), August 1999.

[CRAG00] P. Calhoun, A. Rubens, H. Akhtar, E. Guttman. DIAMETER Base Protocol. IETF Internet Draft, June 2000.

[CTS95] B. Cox, J. Tygar, and M. Sirbu. NetBill security and transaction protocol. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, pp.77-88, New York, July 1995.

[CW99] M. Chan and T. Woo. Next-generation wireless data services: architecture and experience. *IEEE Personal Communications*, 6(1):20-33, February 1999.

[Cyb97] CyberCoin. In *Electronic payment systems* [OPT97], pp. 181-2.

[DA99] T. Dierks and C. Allen. The TLS Protocol version 1.0. Internet Network Working Group, Standards Track, Request for Comments: RFC 2246, January 1999.

[Dai99] W. Dai. Crypto++: a C++ class library of cryptographic primitives, version 3.1, April 1999. Available from ftp://ftp.funet.fi/pub/crypt/cryptography/libs/

[Dam88] I. Damgard. Payment systems and credential mechanisms with provable security against abuse by individuals. In *Advances in Cryptology – CRYPTO '88 Proceedings*, pp. 328-35, Lecture Notes in Computer Science vol. 403. Springer-Verlag, Berlin, 1988.

[DAR81] Defense Advanced Research Projects Agency (DARPA). Internet protocol DARPA Internet program protocol. Request for Comments: RFC 791, September 1981.

[Dav96] G. Davies. *A history of money from ancient times to the present day*. University of Wales Press, Cardiff, 1996.

[DB99] N. Daswani and D. Boneh. Experimenting with electronic commerce on the PalmPilot. In *Financial Cryptography '99 Proceedings*, pp.1-16, Lecture Notes in Computer Science vol. 1648. Springer-Verlag, Berlin, 1999.

[DFTY97] G. Davida, Y. Frankel, Y. Tsiounis, and M. Yung. Anonymity control in e-cash systems. In *Financial Cryptography '97 Proceedings*, pp. 1-16, Lecture Notes in Computer Science vol. 1318. Springer-Verlag, Berlin, 1997.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644-54, November 1976.

[DH95] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. Request for Comments: RFC 1883, December 1995.

[Die97] S. Dietrich. A formal analysis of the secure sockets layer protocol. Ph.D thesis, Department of Mathematics and Computer Science, Adelphi University, New York, April 1997.

[Dig00] Digianswer. Digianswer Bluetooth demo card – Bluetooth neighborhood guide, version 4.11, Nibe, Denmark, 2000.

[DL99] X. Dai and B. Lo. Netpay – an efficient protocol for micropayments on the WWW. In *Proceedings of the 5th Australian World Wide Web Conference (AusWeb'99)*, Southern Cross University, Lismore, Australia, April 1999.

[DMAD00] S. Das, A. Misra, P. Agrawal, and S. Das. TeleMIP: telecommunications-enhanced mobile IP architecture for fast intradomain mobility. *IEEE Personal Communications*, 7(4):50-8, August 2000.

[Dob96] H. Dobbertin. The status of MD5 after a recent attack. *RSA Laboratories' CryptoBytes*, 2(2):1-6, 1996.

[Dra00] J. Dray. NIST performance analysis of the final round Java AES candidates. In *Proceedings of the 3rd AES Candidate Conference*, New York, April 2000.

[EBCY96] D. Eastlake 3rd, B. Boesch, S. Crocker and M. Yesil. CyberCash credit card protocol version 0.8. Internet Network Working Group, Request for Comments: RFC 1898, February 1996.

[EGM89] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In *Advances in Cryptology – CRYPTO '89 Proceedings*, pp. 263-75, Lecture Notes in Computer Science vol. 435. Springer-Verlag, Berlin, 1990.

[EGY83] S. Even, O. Goldreich, and Y. Yacobi. Electronic wallet. In *Advances in Cryptology – CRYPTO '83 Proceedings*, pp. 383-86. Plenum Publishing, New York, 1983.

[EKW74] A. Evans, W. Kantrowitz, and E. Weiss. A user authentication scheme not requiring secrecy in the computer. *Communications of the ACM*, 17(8):437-42, August 1974.

[EMV98] Europay, MasterCard and Visa Corporation. *EMV '96 – integrated circuit card specification for payment systems.* Book 1: Integrated Circuit Card Specification, Book 2: Integrated Circuit Card Terminal Specification, Book 3: Integrated Circuit Card Application Specification, version 3.1.1, May 1998.

[EMV00] Europay, MasterCard and Visa Corporations. *EMV 2000 – integrated circuit card (ICC) specification for payment systems*. Book 1: Application Independent ICC to Terminal Interface Requirements, Book 2: Security and Key Management, Book 3: Application Specification, Book 4: Cardholder, Attendant Interface Requirements, Draft Version 4.0, April 2000.

[EN98] K. Ezawa and G. Napiorkowski. Assessment of threats for smart card based electronic cash. In *Financial Cryptography '98 Proceedings*, pp. 58-72, Lecture Notes in Computer Science vol. 1465. Springer-Verlag, Berlin, 1998.

[Eng98] R. England. Payoff deferred. *Banking Strategies*, 76(2), March 1998.

[ENK99] K. Ezawa, G. Napiorkowski, and M. Kossarski. Assessment of effectiveness of counterfeit transaction detection systems for smart card based electronic cash. In *Financial Cryptography '99 Proceedings*, pp.72-85, Lecture Notes in Computer Science vol. 1648. Springer-Verlag, Berlin, 1999.

[EO94] T. Eng and T. Okamoto. Single-term divisible electronic coins. In *Advances in Cryptology – EUROCRYPT '94 Proceedings*, pp. 306-19, Lecture Notes in Computer Science vol. 576. Springer-Verlag, Berlin, 1994.

[ES97] Elsevier Science. Telecoms fraud in the cellular market: how much is hype and how much is real? *Computer Fraud & Security*, x(x):11-14, June 1997.

[ESTO99] European Science and Technology Observatory (ESTO). *Electronic payment systems in European countries – country synthesis report*, European Commission, Brussels, 1999.

[ETSI95] ETSI ETR 065: Universal personal telecommunication (UPT); requirements on charging, billing and accounting, June 1995.

[ETSI96a] ETSI ETR 123: Broadband integrated services digital network (B-ISDN); parameters and mechanisms provided by the network relevant for charging in B-ISDN. March 1996.

[ETSI96b] ETSI GSM 01.02: Digital cellular telecommunications system (phase 2+); general description of a GSM public land mobile network (PLMN), March 1996.

[ETSI96c] ETSI ETS 300 175-1: Digital enhanced cordless telecommunications (DECT); common interface (CI); part 1: overview, September 1996.

[ETSI96d] ETSI ETS 300 652: Radio equipment and systems (RES); high performance radio local area network (HIPERLAN) type 1; functional specification, October 1996.

[ETSI98a] ETSI TR 101 619: Network aspects (NA); considerations on network mechanisms for charging and revenue accouting, version 1.1.1, November 1998.

[ETSI98b] ETSI TR 101 617: Network aspects (NA); considerations on network mechanisms for charging and revenue accounting for European telephony numbering space (ETNS) services, version 1.1.1, December 1998.

[ETSI98c] ETSI I-ETS 300 819: Telecommunications management network (TMN); functional specification of the usage metering information management on the operations system/network element interface, June 1998.

[ETSI99a] ETSI TR 101 734: Internet protocol (IP) based networks; parameters and mechanisms for charging, version 1.1.1, September 1999.

[ETSI99b] ETSI TS 101 300: Telecommunications and Internet protocol harmonization over networks (TIPHON); Description of technical issues, version .2.1.1, October 1999.

[ETSI99c] ETSI TS 100 616: Digital cellular telecommunications system (phase 2+); event and call data, version 7.0.1, July 1999.

[ETSI00a] ETSI TS 122 115: Digital cellular telecommunications system (phase 2+) (GSM); universal mobile telecommunications system (UMTS); charging and billing, version 3.2.0, January 2000.

[ETSI00b] ETSI TR 122 924: Universal mobile telecommunications system (UMTS); service aspects; charging and accounting mechanisms, version 3.1.1, January 2000.

[ETSI00c] ETSI TS 101 393: Digital cellular telecommunications system (phase 2+); general packet radio service (GPRS); GPRS charging, version 7.5.0, June 2000.

[ETSI00d] ETSI TS 101 321: Telecommunications and Internet protocol harmonization over networks (TIPHON); open settlement protocol (OSP) for inter-domain pricing, authorization, and usage exchange, version 2.1.0, May 2000.

[ETSI00e] ETSI GSM 03.60: Digital cellular telecommunications system (phase 2+); general packet radio service (GPRS); service description; stage 2, version 7.4.0, April 2000.

[ETSI00f] ETSI TS 132 015: Digital cellular telecommunications system (phase 2+) (GSM); universal mobile telecommunications system (UMTS); GSM call and event data for the packet switched domain, version 3.1.1, March 2000.

[ETSI00g] ETSI TS 122 078: Digital cellular telecommunications system (phase 2+) (GSM); universal mobile telecommunications system (UMTS); customised applications for mobile network enhanced logic (CAMEL); service description, stage 1, version 3.3.0, March 2000.

[ETSI00h] ETSI TS 123 057: Digital cellular telecommunications system (phase 2+) (GSM); universal mobile telecommunications system (UMTS); mobile station application execution environment (MExE); functional description; Stage 2, version 3.2.0, June 2000.

[ETSP00] R. Ekstein, Y. T'Joens, B. Sales, and O. Paridaens. AAA protocols : comparison between RADIUS, DIAMETER and COPS. IETF Internet Draft, April 2000.

[EV99] EuroPay and Visa Corporations. *Common electronic purse specifications*. Book 1: Business Requirements, Book 2: Functional Requirements, Book 3: Technical Specifications, September 1999.

[Eve97] D. Everett. Cowry shell to smart card electronic cash. IEE Review, 43(2):59-62, March 1997.

[Fan97] C. Fancher. In your pocket: smartcards. *IEEE Spectrum*, 34(2):47-53, February 1997.

[FB96] N. Freed and N. Borenstein. Multipurpose Internet mail extensions (MIME) part one: format of Internet message bodies. Request for Comments: RFC 2045, November 1996.

[FCL99] Y. Fang, I. Chlamtac, Y.Lin. Billing strategies and performance analysis for PCS networks. *IEEE Transactions on Vehicular Technology*, 48(2):638-51, March 1999.

[Fer90] J. Domingo-Ferrer. Software run-time protection: a cryptographic issue. In *Advances in Cryptology – EUROCRYPT '90 Proceedings*, pp. 474-80, Lecture Notes in Computer Science vol. 473. Springer-Verlag, Berlin, 1990.

[Fer93a] N. Ferguson. Single term off-line coins. In *Advances in Cryptology – EUROCRYPT '93 Proceedings*, pp. 318-28, Lecture Notes in Computer Science vol. 765. Springer-Verlag, Berlin, 1993.

[Fer93b] N. Ferguson. Extensions of single-term coins. In *Advances in Cryptology – CRYPTO '93 Proceedings*, pp. 302-18, Lecture Notes in Computer Science vol. 773. Springer-Verlag, Berlin, 1993.

[FH00] S. Farrell and R. Housley. An Internet attribute certificate profile for authorization. IETF Internet Draft, May 2000.

[FHJ98] J. Francis, H. Herbrig, and N. Jefferies. Secure provision of UMTS services over diverse access networks. *IEEE Communications Magazine*, 36(2):128-36, February 1998.

[FJ98] E. Franz and A. Jerichow. A mix-mediated anonymity service and its payment, *Computer Security - ESORICS '98 Proceedings*, pp. 313-27, Lecture Notes in Computer Science vol. 1485. Springer-Verlag, Berlin, 1998.

[FJ99] J. Domingo-Ferrer and J. Herrera-Joancomarti. Spending programs: a tool for flexible micropayments. In *Proceedings of Information Security Workshop (ISW'99, Kuala Lumpur, Malaysia)*, pp.1-13, Lecture Notes in Computer Science vol. 1729. Springer-Verlag, Berlin, November 1999.

[FJW98] E. Franz, A. Jerichow, and G. Wicke. A payment scheme for mixes providing anonymity. In *Trends in Distributed Systems for Electronic Commerce (TREC'98) Conference Proceedings*, pp.94-108, Lecture Notes in Computer Science vol. 1402. Springer-Verlag, Berlin, 1998.

[FKK96] A. Frier, P. Karlton, and P. Kocher. The SSL 3.0 protocol. Netscape Communications Corporation, November, 1996.

[FL98] B. Fox and B. LaMacchia. Certificate revocation: mechanics and meaning. In *Financial Cryptography '98 Proceedings*, pp. 158-64, Lecture Notes in Computer Science vol. 1465. Springer-Verlag, Berlin, 1998.

[FL99] B. Fox and B. LaMacchia, Online certificate status checking in financial transactions: the case for re-issuance. In *Financial Cryptography '99 Proceedings*, pp.104-17, Lecture Notes in Computer Science vol. 1648. Springer-Verlag, Berlin, 1999.

[Flo97] J. Flood. *Telecommunication networks*. 2nd edition, Institution of Electrical Engineers, London, 1997.

[FMMO99] A. Furuskar, S. Mazur, F. Muller, and H. Olofsson. EDGE: enhanced data rates for GSM and TDMA/136 evolution. *IEEE Personal Communications*, 6(3):56-66, June 1999.

[FNO99] A. Furuskar, J. Naslund, and H. Olofsson. EDGE – enhanced data rates for GSM and TDMA/136 evolution. *Ericsson Review*, 76(1):28-37, 1999.

[FRGH+99] A. Fasbender, F. Reichert, E. Geulen, J. Hjelm, and T. Wierlemann. Any network, any terminal, anywhere. *IEEE Personal Communications*, 6(2):22-30, April 1999.

[FTC98] U.S. Federal Trade Commission. Truth-in-billing and billing format. Federal Communications Commission Comment, CC Docket No. 98-170, November 1998.

[FTY96] Y. Frankel, Y. Tsiounis, and M. Yung. Indirect discourse proofs: achieving efficient fair off-line e-cash. In *Advances in Cryptology – ASIACRYPT '96 Proceedings*, pp. 286-300, Lecture Notes in Computer Science vol. 1163. Springer-Verlag, Berlin, 1996.

[FW96] A. Furche and G. Wrightson. SubScrip - an efficient payment mechanism for pay-per-view services on the Internet. In *Proceedings of the 5th IEEE International Conference for Computer Communication and Networks*, Maryland, October 1996.

[FY95] Y. Frankel and M. Yung. Escrow encryption systems visited: attacks, analysis and designs. In *Advances in Cryptology – CRYPTO '95 Proceedings*, pp. 222-35, Lecture Notes in Computer Science vol. 963. Springer-Verlag, Berlin, 1995.

[GA97] GSM Association PRD TG24: Requirements for charging, billing, accounting and tariffing, version 3.0.0, October 1997.

[Gem98] Gemplus. GemXpresso tutorial. Gemenos Cedex, France, September 1998.

[GHJP00] S. Glass, T. Hiller, S. Jacobs, and C. Perkins. Mobile IP authentication, authorization, and accounting requirements. IETF Internet Draft, March 2000.

[GJS96] J. Gosling, B. Joy, and G. Steele. *The Java language specification*. Addison-Wesley, Massachusetts, 1996.

[GMA+95] S. Glassman, M. Manasse, M. Abadi, P. Gauthier and P. Sobalvarro. The Millicent protocol for inexpensive electronic commerce. In *Proceedings of the 4th International World Wide Web Conference, World Wide Web Journal*, 1(1):603-18. O'Reilly & Associates, December 1995.

[GR98] R. Grehan and D. Rowell. jBYTEMark Java benchmark software. BYTE Magazine, McGraw-Hill, New York, May 1998.

[Gri00] F. Grieu. A chosen message attack on the ISO/IEC 9796-1 signature scheme. In *Advances in Cryptology – EUROCRYPT '2000 Proceedings*, pp.70-80, Lecture Notes in Computer Science vol. 1807. Springer-Verlag, Berlin, 2000.

[GS96] E. Gabber and A. Silberschatz. Agora: a minimal distributed protocol for electronic commerce. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pp.223-32, Oakland, California, November 1996.

[GSMA98] GSM Association PRD BA.12: Transferred account procedure and billing information, August 1998.

[GSMA00] GSM Association PRD TD.57: Transferred account procedure version 3 (TAP3), March 2000.

[GW99] H. Granbohm and J. Wiklund. GPRS – general packet radio service. *Ericsson Review*, 76(2):82-8, 1999.

[Haa98] J. Haartsen. Bluetooth - the universal radio interface for ad hoc, wireless connectivity. *Ericsson Review*, 75(3):110-7, 1998.

[Haa00] J. Haartsen. The Bluetooth radio system. *IEEE Personal Communications*, 7(1):28-36, February 2000.

[Hal94] N. Haller. The S/KEY one-time password system. In *Proceedings of the 1994 Internet Society (ISOC) Symposium on Network and Distributed System Security*, pp.151-57, San Diego, California, February 1994.

[Hal95] N. Haller. The S/KEY one-time password system. Request for Comments: RFC 1760, February 1995.

[HBFH97] V. Hassler, R. Bihlmeyer, M. Fischer, and M. Hauswirth. MiMi: a Java implementation of the MicroMint scheme. In *Proceedings of the 2nd World Conference of the WWW, Internet, and Intranet (WebNet '97)*, Toronto, Canada, November 1997.

[HE98] J. Hickie and R. Evans. Application of neural networks to telecommunications fraud detection. Broadcom Communicate 4(1), Broadcom Eireann Research, Dublin, June 1998.

[Hei99] J. Heiss. The network is the car. Sun Microsystems, Palo Alto, California, June 1999.

[Hen97] M. Hendry. *Smart card security and applications.* Artech House, Boston/London, 1997.

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 public key infrastructure certificate and CRL profile. Request for Comments: RFC 2459, January 1999.

[HHMM+98] G. Horn, P. Howard, K. Martin, C. Mitchell, B. Preneel and K. Rantos. Trialling secure billing with trusted third party support for UMTS applications. In *Proceedings of the 3rd ACTS Mobile Communication Summit*, Vol 2, pp 574-79, Rhodes, Greece, June 1998.

[Hil95] S. Hild. A brief history of mobile telephony. University of Cambridge Computer Laboratory Technical Report TR372, Cambridge, UK, January 1995.

[Hil99a] T. Hiller (editor). 3G wireless data provider architecture using mobile IP and AAA. IETF Internet Draft, March 1999.

[Hil99b] T. Hiller (editor). CDMA2000 wireless data requirements for AAA. IETF Internet Draft, November 1999.

[HMN+98] N. Haller, C. Metz, P. Nesser, and M. Straw. A one-time password system. Request for Comments: RFC2289, February 1998.

[HMV99] G. Horn, K. Muller, and B. Vinck. Towards a UMTS security architecture. In *Proceedings of European Wireless '99*, pp. 495-500, Munich, October 1999.

[HP98] G. Horn and B. Preneel. Authentication and payment in future mobile systems. *In Proceedings of Computer Security – ESORICS '98*, pp. 277-93, Lecture Notes in Computer Science vol. 1485. Springer-Verlag, Berlin 1998.

[HP99] Hewlett-Packard. Smart Internet usage – managing service usage data for strategic advantage. White Paper, April 1999.

[HPT97] R. Hauser, A. Przygienda, and G. Tsudik. Reducing the cost of security in link state routing. In *Proceedings of the 1997 Internet Society (ISOC) Symposium on Network and Distributed System Security*, pp.93-9, San Diego, California, February 1997.

[HPT99] R. Hauser, A. Przygienda, and G. Tsudik. Lowering security overhead in link state routing. *Computer Networks –The International Journal of Distributed Informatique*, 31(8):885-94, April 1999.

[HSW96] R. Hauser, M. Steiner, and M. Waidner. Micro-payments based on iKP. In *Proceedings of the 14th Worldwide Congress on Computer and Communications Security Protection*, pp. 67-82, Paris, 1996.

[HT96] R. Hauser and G. Tsudik. On shopping incognito. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pp.251-7, Oakland, California, November 1996.

[HY97] A. Herzberg and H. Yochai. Mini-Pay: charging per click on the Web. In *Proceedings of the 6th International World Wide Web Conference*, Santa Clara, California, April 1997.

[HZI98] G. Hanaoka, Y. Zheng, and H. Imai. LITESET: A lightweight secure electronic transaction protocol. In *Proceedings of the 3rd Australasian Conference on Information Security and Privacy (ACISP'98)*, pp.215-26, Lecture Notes in Computer Science vol. 1438. Springer-Verlag, Berlin, 1998.

[IEEE99] IEEE 802-11: Information technology – telecommunications and information exchange between systems – local and metropolitan area networks – specific requirements – part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications. 1999.

[Inf99] Infineon Technologies. Security and chip card ICs, SLE 66CX320P technical specification. Munich, July 1999.

[IPDR00] IPDR Organization. Network data management – usage (NDM-U) for IP-based services, version 1.1, June 2000.

[ISO90] ISO/IEC 8824: Information technology – open systems interconnection – specification of abstract syntax notation one (ASN.1), 1990.

[ISO91] ISO/IEC 9796: Information technology – security techniques – digital signature scheme giving message recovery, July 1991.

[ISO94] ISO/IEC 10118-2: Information technology - security techniques - hash-functions - part 2: hash-functions using an n-bit block cipher algorithm, 1994.

[ISO95] ISO/IEC 7816-4: Integrated circuit cards with contacts – part 4: interindustry commands for interchange, 1995.

[ISO97] ISO/IEC 7816-3: Integrated circuit cards with contacts – part 3: electronic signals and transmission protocols, 1997.

[ISO99a] ISO/IEC JTC 1/SC 27. SC 27N 2424: Recommendation on the withdrawal of IS 9796:1991, October 1999.

[ISO99b] ISO 9594-8 information technology - open systems interconnection - the directory: authentication framework - draft amendment 1: certificate extensions. October 1999.

[ITU97a] ITU-T Recommendation X.509: the directory: authentication framework, August 1997.

[ITU97b] ITU-T Recommendation E.164: The international public telecommunication numbering plan, May 1997.

[ITU98a] ITU Q.825: Draft specification of TMN applications at the Q3 interface: call detail recording, June 1998.

[ITU98b] ITU-T Recommendation D.140: Accounting rate principles for international telephone services, July 1998.

[ITU99a] ITU-T Recommendation H.323: Packet based multimedia communications systems, September 1999.

[ITU99b] ITU-T Recommendation Z.100. Languages for telecommunications applications – specification and description language, November 1999.

[JB95] M. Jones and B. Schneier. Securing the World Wide Web: smart tokens and their implementation. In *Proceedings of the 4th International World Wide Web Conference, World Wide Web Journal*, 1(1):397-409, December 1995.

[JJ99] M. Jakobsson and A. Juels. Proofs of work and bread pudding protocols. In *Secure Information Networks - Communications and Multimedia Security - CMS '99 Proceedings*. Kluwer Academic Publishers, Boston, September 1999.

[JLO97] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology – CRYPTO '97 Proceedings*, pp. 150-64, Lecture Notes in Computer Science vol. 1294. Springer-Verlag, Berlin, 1997.

[JM97] A. Jurisic and A. Menezes. Elliptic curves and cryptography. Certicom White Paper, Ontario, Canada, November 1997.

[JO97] S. Jarecki and A. Odlyzko. An efficient micropayment system based on probabilistic polling. In *Financial Cryptography '97 Proceedings*, pp. 173-91, Lecture Notes in Computer Science vol. 1318. Springer-Verlag, Berlin, 1997.

[JY96] C. Jutla and M. Yung. PayTree: amortized-signature for flexible micropayments. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pp.213-21, Oakland, California, November 1996.

[JY97] M. Jakobsson and M. Yung. Applying anti-trust policies to increase trust in a versatile e-money system. In *Financial Cryptography '97 Proceedings*, pp. 217-38, Lecture Notes in Computer Science vol. 1318. Springer-Verlag, Berlin, 1997.

[Kal98] B. Kaliski. Recommendations on elliptic curve cryptosystems. RSA Laboratories Technical Note, Bedford, Massachusetts, March 1998.

[Kal00] O. Kallstrom. Business solutions for mobile e-commerce. *Ericsson Review*, 77(2):80-92, 2000.

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: keyed-hashing for message authentication. Request for Comments: RFC 2104, February 1997.

[Kha99] R. Khare. W* effect considered harmful. *IEEE Internet Computing*, 3(4):89-92, July 1999.

[KH00] U. Kruck and S. Heng. M-commerce: mega business or Mickey Mouse? *Deutsche Bank Research Economics*, 1(8):3-11, October 2000.

[KLM94] D. Kristol, S. Low, and N. Maxemchuk. Anonymous Internet mercantile protocol. AT&T Bell Laboratories, Murray Hill, New Jersey, March 1994.

[KMM00] R. Kalden, I. Meirick, and M. Meyer. Wireless Internet access based on GPRS. *IEEE Personal Communications*, 7(2):8-18, April 2000.

[Koc98] P. Kocher. On certificate revocation and validation. In *Financial Cryptography '98 Proceedings*, pp. 172-77, Lecture Notes in Computer Science vol. 1465. Springer-Verlag, Berlin, 1998.

[KOO99] B. Kerf, S. Overbeek, and C. Otten. N-Count – fast, low-cost secure payments using smart cards. QC Technology BV, The Netherlands, June 1999.

[KR95] B. Kaliski and M.Robshaw, Message authentication with MD5. *RSA Laboratories' CryptoBytes*, 1(1):5-8, 1995.

[Kra99] H. Krawczyk. Blinding of credit card numbers in the SET protocol. In *Financial Cryptography '99 Proceedings*, pp.17-28, Lecture Notes in Computer Science vol. 1648. Springer-Verlag, Berlin, 1999.

[Lam81] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770-72, November 1981.

[Lan98] P. Landrock. TTPs overview – concepts and review of state of art from a technical point of view. In *State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography - Revised Lectures*, pp.241-63, Lecture Notes in Computer Science vol. 1528. Springer-Verlag, Berlin, 1998.

[Lar00] J. Larmouth. *ASN.1 complete*. Morgan Kaufmann, San Diego/London, 2000.

[LCR00] Y. Lin, M. Chang, and H. Rao. Mobile prepaid phone services. *IEEE Personal Communications*, 7(3):6-14, June 2000.

[LCW97] D. Lam, D. Cox, and J. Widom. Teletraffic Modeling for Personal Communications Services. *IEEE Communications*, 35(2):79-87, February 1997.

[LGGV+00] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. Generic AAA Architecture. IETF Internet Draft, March 2000.

[LGLK+96] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones. SOCKS protocol version 5. Request for Comments: RFC 1928, March 1996.

[Li99] T. Li. MPLS and the evolving Internet architecture. *IEEE Communications*, 37(12):38-41, December 1999.

[LMP94] S. Low, N. Maxemchuk, and S. Paul. Anonymous credit cards. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 108-17, Fairfax, Virginia, November 1994.

[LMP96] S. Low, N. Maxemchuk, and S. Paul. Anonymous credit cards and their collusion analysis. *IEEE/ACM Transactions on Networking*, 4(6):809-16, December 1996.

[LO98] R. Lipton and R. Ostrovsky. Micro-payments via efficient coin-flipping. In *Financial Cryptography '98 Proceedings*, pp. 1-15, Lecture Notes in Computer Science vol. 1465. Springer-Verlag, Berlin, 1998.

[LV00] A. Lenstra and E. Verheul. The XTR public key system. In *Advances in Cryptology – CRYPTO 2000 Proceedings*, pp. 1-19, Lecture Notes in Computer Science vol. 1880. Springer-Verlag, Berlin, 2000.

[LY95] P. Lisiecki and Y. Yerushalmi. PayMe! A customer-oriented efficient micropayment scheme. Massachusetts Institute of Technology, Cambridge, Massachusetts, December 1995.

[MAMG+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet public key infrastructure online certificate status protocol – OCSP. Request for Comments: RFC 2560, June 1999.

[Man95] M. Manasse. The Millicent protocols for electronic commerce. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*. pp.117-23, New York, USA, July 1995.

[Man98] Mandate II Consortium. Mandate final report. European Communities DGXIII Electronic Trusted Services Programme, February 1998.

[Mao96] W. Mao. Lightweight micro-cash for the Internet. In *Computer Security - ESORICS '96 Proceedings*, pp. 15-32, Lecture Notes in Computer Science vol. 1146. Springer-Verlag, Berlin, 1996.

[MAO00] MAOSCO Consortium. A guide to the MULTOS scheme, version 2.0. London, March 2000.

[Mas00] MasterCard. *MasterCard chip (M/Chip) for debit and credit*, Book 1: Business Functional Requirements, Book 2: Recommended Specifications, Book 3: Minimum Card Requirements, Book 4: Terminal Requirements, Book 5: Personalization Data Specifications, Version 2.1, January 2000.

[Mat91] S. Matyas. Key processing with control vectors. Journal of Cryptology, 3(2):113-36, 1991.

[MBLT+99] J. Mitola, V. Bose, B. Leiner, T. Turletti, and D. Tennenhouse (Editors). Special issue on software radios. *IEEE Journal on Selected Areas in Communications*, 17(4), April 1999.

[MEH00] L. Mathy, C. Edwards, and D. Hutchison. The Internet: a global telecommunications solution? *IEEE Network*, 14(4): 46-57, July 2000.

[Mer87] R. Merkle. A digital signature based on conventional encryption function. In *Advances in Cryptology – CRYPTO '87 Proceedings*, pp. 369-78, Lecture Notes in Computer Science vol. 293. Springer-Verlag, Berlin, 1988.

[Mer89] R. Merkle. A certified digital signature. In *Advances in Cryptology – CRYPTO '89 Proceedings*, pp. 218-38, Lecture Notes in Computer Science vol. 435. Springer-Verlag, Berlin, 1990.

[Met99] C. Metz. AAA protocols: authentication, authorization, and accounting for the Internet. IEEE Internet Computing, 3(6):75-79, November 1999.

[Mey70] P. Meyer. *Introducing probability and statistical applications*, 2nd Edition, Addison-Wesley, London, 1970.

[MH00] P. McCann and T. Hiller. An Internet infrastructure for cellular CDMA networks using mobile IP. *IEEE Personal Communications*, 7(4):26-32, August 2000.

[Mic96] S. Micali. Efficient certificate revocation. Technical Memo MIT/LCS/TM-542b, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, March 1996.

[Mic98a] P. Michon. Payment in electronic commerce. In *Proceedings of the 8th Internet Society Conference on Internet Networking (INET'98)*, Geneva, Switzerland, July 1998.

[Mic98b] S. Micali. Certificate revocation system. U.S. Patent 5,793,868, U.S. Department of Commerce, August 1998.

[Mil85] V. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology – CRYPTO '85 Proceedings*, pp. 417-26, Lecture Notes in Computer Science vol. 218. Springer-Verlag, Berlin, 1985.

[Miy98] H. Miyano. One-time digital signature and pseudo k-time digital signature. *IEICE Transactions on Fundamentals of Electronics Communications & Computer Sciences*, E81-A(1):48-55, Japan, January 1998.

[MMS97] Y. Matias, A. Mayer, and A. Silberschatz. Lightweight security primitives for e-commerce. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pp.95-102, Monterey, California, December 1997.

[MN93] G. Medvinsky and B. Clifford Neuman. NetCash: a design for practical electronic currency on the Internet. In *Proceedings of 1st the ACM Conference on Computer and Communication Security*, November 1993.

[MNVB+99] J. Maassen, R. Nieuwpoort, R. Veldema, H. Bal, and A. Plaat. An efficient implementation of Java's remote method invocation. In *Proceedings of the 7th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 173-82, Atlanta, Georgia, May 1999.

[MOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. CRC Press, Boca Raton, 1996.

[MPMH+98] K. Martin, B. Preneel, C. Mitchell, H. Hitz, G. Horn, A. Poliakova, and P. Howard. Secure billing for mobile information services in UMTS. *In Proceedings of Intelligence in Services and Networks (IS&N '98)*, pp. 535-48, Lecture Notes in Computer Science vol. 1430. Springer-Verlag, Berlin 1998.

[MR00] P. McDaniel and A. Rubin. A response to "can we eliminate certificate revocation lists?" To appear in *Financial Cryptography '00 Proceedings*, Anguilla, British West Indies, February 2000.

[MRa96] D. M'Raihi. Cost-effective payment schemes with privacy regulation. In *Advances in Cryptology – ASIACRYPT '96 Proceedings*, pp. 266-75, Lecture Notes in Computer Science vol. 1163. Springer-Verlag, Berlin, 1996.

[Mur00] J. Murphy. Assuring performance in e-commerce systems. In Proceedings of the 16th IEE UK Teletraffic Symposium, Harlow, England, May 2000.

[MV97] MasterCard and VISA Corporations. *SET secure electronic transaction specification, version 1.0*. Book 1: Business Description, Book 2: Programmer's Guide, Book 3: Formal Protocol Definition, May 1997.

[MV98a] Y. Mu and V. Varadharajan. Anonymous Internet credit cards. In *Proceedings of the 21st National Information Systems Security Conference (NISSC'98)*, pp229-37, Crystal City, Virginia, October 1998.

[MV98b] Y. Mu and V. Varadharajan. A new scheme of credit based payment for electronic commerce. In *Proceedings of the 23rd Annual Conference on Local Computer Networks (LCN'98)*, pp.278-84. IEEE Computer Society, Boston, October 1998.

[MVL97] Y. Mu, V. Varadharajan, and Y Lin. New micropayment schemes based on PayWords. In *Proceedings of the 2nd Australasian Conference on Information Security and Privacy (ACISP'97)*, pp. 283-93, Lecture Notes in Computer Science vol. 1270. Springer-Verlag, Berlin, 1997.

[Mye98] M. Myers. Revocation: options and challenges. In *Financial Cryptography '98 Proceedings*, pp. 165-71, Lecture Notes in Computer Science vol. 1465. Springer-Verlag, Berlin, 1998.

[Nef99] J. Neffenger. VolanoMark Java benchmark software, version 2.1. Volano LLC, San Francisco, California, 1999. Available from http://www.volano.com/

[NIST94] NIST Federal Information Processing Standards Publication (FIPS) 186: Digital signature standard (DSS). U.S. Department of Commerce, May 1994.

[NIST95] NIST Federal Information Processing Standards Publication (FIPS) 180-1: Secure hash standard (SHS). U.S. Department of Commerce, April 1995.

[NIST99] NIST Federal Information Processing Standards Publication (FIPS) 46-3: Data encryption standard (DES). U.S. Department of Commerce, October 1999.

[NM95] C. Neuman and G. Medvinsky. Requirements for network payment: the NetCheque perspective. In *Proceedings of IEEE Compcon'95*. pp. 32-6, San Francisco, March 1995.

[NN98] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium*, pp.217-28, San Antonio, Texas, January 1998.

[NN00] M. Naor and K. Nissim. Certificate revocation and certificate update. *IEEE Journal on Selected Areas in Communications*, 18(4):561-70, April 2000.

[NPH99] C. Nester, M. Philippsen and B. Haumacher. A more efficient RMI for Java. In *Proceedings of the 1999 ACM conference on Java Grande*, pp. 152-9, Palo Alto, California, June 1999.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993-9, December 1978.

[NSL00] K. Negus, A. Stephens, and J. Lansford. HomeRF: wireless networking for the connected home. *IEEE Personal Communications*, 7(1):20-7, February 2000.

[NT94] C. Neuman and T. Ts'o. Kerberos: an authentication service for computer networks. *IEEE Communications*, 32(9):33-8, September 1994.

[ODTP98] D. O'Mahony, L. Doyle, H. Tewari, and M. Peirce. An application to provide UMTS telephony services on fixed terminals in COBUCO. In *Proceedings of the 3rd ACTS Mobile Communications Summit*, pp.72-6, Rhodes, Greece, June 1998.

[OFPR+94] A. Olsen, O. Faergemand, B. Pedersen, R. Reed, and J. Smith. *Systems engineering using SDL-92*. Elsevier, Amsterdam/Oxford, 1994.

[OFTL00] OFTEL. Ensuring telephone bills are accurate – a statement on the review of the OFTEL meter approval scheme for public telecommunications operators. February 2000.

[Oka95] T. Okamoto. An efficient divisible electronic cash scheme. In *Advances in Cryptology – CRYPTO '95 Proceedings*, pp. 438-51, Lecture Notes in Computer Science vol. 963. Springer-Verlag, Berlin, 1995.

[Oli96] D. Olivier. The Clickshare service. In *Workshop on Internet Survey Methodology and Web Demographics*, Massachusetts Institute of Technology, Cambridge, Massachusetts, January 1996.

[Oma98] D. O'Mahony. UMTS: The fusion of fixed and mobile networking. *IEEE Internet Computing*, 2(1):49-56, January 1998.

[OO89] T. Okamoto and K. Ohta. Disposable zero-knowledge authentication and their applications to untraceable electronic cash. In *Advances in Cryptology – CRYPTO '89 Proceedings*, pp. 481-96, Lecture Notes in Computer Science vol. 435. Springer-Verlag, Berlin, 1990.

[OO91] T. Okamoto and K. Ohta. Universal electronic cash. In *Advances in Cryptology – CRYPTO '91 Proceedings*, pp. 324-37, Lecture Notes in Computer Science vol. 576. Springer-Verlag, Berlin, 1991.

[OPT97] D. O'Mahony, M. Peirce, and H. Tewari. *Electronic payment systems*. Artech House, Boston/London, 1997.

[OR87] D. Otway and O. Rees. Efficient and timely mutual authentication. *ACM Operating System Review*, 21(1):8-10, January 1987.

[OW91] P. van Oorschot and M. Wiener. A known plaintext attack on two-key triple encryption. In *Advances in Cryptology – EUROCRYPT '90 Proceedings*, pp. 318-25, Lecture Notes in Computer Science vol. 473. Springer-Verlag, Berlin, 1990.

[Pat00] P. Patsuris. New payment systems hope to cash in. *Forbes E-business*, June 9, 2000.

[PD00] G. Patel and S. Dennett. The 3GPP and 3GPP2 movements toward an all-IP mobile network. *IEEE Personal Communications*, 7(4):62-4, August 2000.

[Ped96] T. Pederson. Electronic payments of small amounts. In *Proceedings of the 4th Security Protocols International Workshop (Security Protocols)*, pp. 59-68, Lecture Notes in Computer Science vol. 1189. Springer-Verlag, Berlin, 1996.

[Peh00] S. Pehrson. WAP—The catalyst of the mobile Internet. *Ericsson Review*, 77(1):14-19, 2000.

[Per96] C. Perkins. IP mobility support. Request for Comments: RFC 2002, October 1996.

[Per00] C. Perkins. Mobile IP joins forces with AAA. *IEEE Personal Communications*, 7(4):59-61, August 2000.

[PGH95] J. Padgett, C. Gunther, and T. Hattori. Overview of wireless personal communications. *IEEE Communications*, 33(1):28-41, January 1995.

[PHS98] T. Poutanen, H. Hinton, and M. Stumm. NetCents: a lightweight protocol for secure micropayments. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, pp.25-36, Boston, Massachusetts, September 1998.

[PHS00] P. Pan, E. Hahne and H. Schulzrinne. BGRP: a tree-based aggregation protocol for Inter-domain reservations. *Journal of Communications and Networks*, 2(2):157-67, June 2000.

[PO95] M. Peirce and D. O'Mahony. Scalable secure cash payment for WWW resources with the PayMe protocol set. *World Wide Web Journal*, 1(1):587-602, O'Reilly & Associates, December 1995.

[PO99a] M. Peirce and D. O'Mahony. Multi-party payments for mobile services. Technical Report TCD-CS-1999-48, Dept. of Computer Science, Trinity College Dublin, Ireland, September 1999.

[PO99b] M. Peirce and D. O'Mahony. Micropayments for mobile networks. In *Proceedings of European Wireless '99*, pp. 199-204, Munich, Germany, October 1999.

[PO99c] M. Peirce and D. O'Mahony. Flexible real-time payment methods for mobile communications. *IEEE Personal Communications*, 6(6):44-55, December 1999.

[Pos81] J. Postel. Transmission control protocol - DARPA Internet program protocol specification. Request for Comments: RFC 793, September 1981.

[PPYS+99] C. Polyzois, K. Purdy, P. Yang, D. Shrader, H. Sinnreich, F. Menard, and H. Schulzrinne. From POTS to PANS: a commentary on the evolution to Internet telephony. IEEE Network, 13(3):58-63, May 1999.

[PRA98] B. Preneel, V. Rijmen, and A. Bosselaers. Recent developments in the design of conventional cryptographic algorithms. In *State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography - Revised Lectures*, pp.105-30, Lecture Notes in Computer Science vol. 1528. Springer-Verlag, Berlin, 1998.

[Pre98] B. Preneel. Cryptographic primitives for information authentication – state of the art. In *State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography - Revised Lectures*, pp.49-104. Lecture Notes in Computer Science vol. 1528. Springer-Verlag, Berlin, 1998.

[PS99] P. Pan and H. Schulzrinne. YESSIR: a simple reservation mechanism for the Internet. *Computer Communication Review*, 29(2):89-101, April 1999.

[PW91] B. Pfitzmann and M. Waidner. How to break and repair a "provably secure" untraceable payment system. In *Advances in Cryptology – CRYPTO '91 Proceedings*, pp. 338-50, Lecture Notes in Computer Science vol. 576. Springer-Verlag, Berlin, 1991.

[PW97] B. Pfitzmann and M. Waidner. Strong loss tolerance of electronic coin systems. ACM Transactions on Computer Systems 15(2):194-213, May 1997.

[PW00] I. Pearson and R. Wilson. The future of the phone box. British Telecom Technology Journal, 18(1), Millennium Edition, January 2000.

[Rab78] M. Rabin. Digitalized signatures. In *Foundations of Secure Computation*, R. DeMillo et al. eds, pp. 155-68. Academic Press, New York/London, 1978.

[RAM98] M. Reiter, V. Anupam, and A. Mayer. Detecting hit shaving in click-through payment schemes. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, pp.155-66, Boston, Massachusetts, September 1998.

[Ram99] B. Ramsdell. S/MIME Version 3 Message Specification. Request for Comments: RFC 2633, June 1999.

[RE97] W. Rankl and W. Effing. *Smart card handbook*. John Wiley & Sons, Chichester, USA, 1997.

[Rig97] C. Rigney, RADIUS accounting. Request for Comments: RFC 2139, April 1997.

[Riv92] R. Rivest. The MD5 message-digest algorithm. Request for Comments: RFC 1321, April 1992.

[Riv97] R. Rivest. Electronic lottery tickets as micropayments. In *Financial Cryptography '97 Proceedings*, pp. 307-14, Lecture Notes in Computer Science vol. 1318. Springer-Verlag, Berlin, 1997.

[Riv98] R. Rivest, Can we eliminate certificate revocation lists? In *Financial Cryptography '98 Proceedings*, pp. 178-83, Lecture Notes in Computer Science vol. 1465. Springer-Verlag, Berlin, 1998.

[Roh99] P. Rohatgi. A hybrid signature scheme for multicast source authentication. IETF Internet Draft, June 1999.

[Ros93] K. Rosen. *Elementary number theory and its applications.* Addison-Wesley, Reading, Massachusetts, 1993.

[RPST+00] R. Ramjee, T. La Porta, L. Salgarelli, S. Thuel, K. Varadhan, and L. Li. IP-based access network infrastructure for next-generation wireless data networks. *IEEE Personal Communications*, 7(4):34-41, August 2000.

[RRSW97] C. Rigney, A. Rubens, W. Simpson, and S. Willens. Remote authentication dial in user service (RADIUS). Request for Comments: RFC 2138, April 1997.

[RRSY98] R. Rivest, M. Robshaw, R. Sidney, and Y. Yin. The RC6 block cipher. National Institute of Standards and Technology (NIST) Advanced Encryption Standard (AES) Proposal, June 1998.

[RS96] R. Rivest and A. Shamir. PayWord and MicroMint: two simple micropayment schemes. In *Proceedings of the 4th Security Protocols International Workshop (Security Protocols)*, pp. 69-87, Lecture Notes in Computer Science vol. 1189. Springer-Verlag, Berlin, 1996.

[RS98] A. Romao and M. da Silva. An agent-based secure Internet payment system for mobile computing. In *Trends in Distributed Systems for Electronic Commerce (TREC '98) Conference Proceedings*, pp. 80-93, Lecture Notes in Computer Science vol. 1402. Springer-Verlag, Berlin, 1998.

[RSA99a] RSA Laboratories. PKCS #5 v2.0: Password-based cryptography standard. Bedford, Massachusetts, March 1999.

[RSA99b] RSA Laboratories. PKCS #1 v2.1: RSA cryptography standard. Bedford, Massachusetts, September 1999.

[RSA00] RSA Laboratories. Frequently asked questions about today's cryptography, version 4.1. Bedford, Massachusetts, 2000.

[RV94] F. Redmill and A. Valdar. *SPC digital telephone exchanges.* Institution of Electrical Engineers, London, 1994.

[RWO95] S. Redl, M. Weber, M. Oliphant. *An introduction to GSM*. Artech House, Boston/London, 1995.

[RY97] M. Robshaw and Y. Yin. Elliptic curve cryptosystems. RSA Laboratories Technical Note, Bedford, Massachusetts, June 1997.

[Sam00] H. Samueli. The broadband revolution. IEEE Micro, 20(2):16-26, March 2000.

[Sch89] C. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology – CRYPTO '89 Proceedings*, pp. 239-52, Lecture Notes in Computer Science vol. 435. Springer-Verlag, Berlin, Germany, 1990.

[Sch96] B. Schneier. *Applied cryptography – protocols, algorithms, and source code in C*, 2nd edition. Wiley, New York, 1996.

[Sch98] B. Schoenmakers. Security aspects of the Ecash payment system. In *State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography - Revised Lectures*, pp.338-52, Lecture Notes in Computer Science vol. 1528. Springer-Verlag, Berlin, 1998.

[SET98] Secure Electronic Transaction LLC (SETCo). Generic cryptogram extension, December 1998.

[SET99] Secure Electronic Transaction LLC (SETCo). Common chip extension, September 1999.

[Sey98] B. Seymour. The brain behind fraud control. *Telecommunications International*, 32(11):71-2, Horizon House Publications, November 1998.

[SFPW98] B. Stiller, G. Fankhauser, B. Plattner, and N. Weiler. Charging and accounting for Internet services – state of the art, problems, and trends. In *Proceedings of the 8th Internet Society Conference on Internet Networking (INET'98)*, Geneva, Switzerland, July 1998.

[SH97] E. Solana and J. Harms. Flexible Internet secure transactions based on collaborative domains. In *Proceedings of the 5th Security Protocols International Workshop (Security Protocols)*, pp. 37-51, Lecture Notes in Computer Science vol. 1361. Springer-Verlag, Berlin, 1997.

[Sim96] W. Simpson. PPP challenge handshake authentication protocol (CHAP). Request for Comments: RFC 1994, August 1996.

[Sir97] M. Sirbu. Credits and debits on the Internet. *IEEE Spectrum 34(2):23-9*, February 1997.

[SKWW+99] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Performance comparison of the AES submissions. In *Proceedings of the 2nd AES Candidate Conference*, Rome, Italy, March 1999.

[SL00] A. Sterbenz and P. Lipp. Performance of the AES candidate algorithms in Java. In *Proceedings of the 3rd AES Candidate Conference*, New York, April 2000.

[SN92] B. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers and Security*, 11(6):581-3, October 1992.

[SPC95] M. Stadler, J. Piveteau, and J. Camenisch. Fair blind signatures. In *Advances in Cryptology – EUROCRYPT '95 Proceedings*, pp. 209-19, Lecture Notes in Computer Science vol. 921. Springer-Verlag, Berlin, 1995.

[ST95] M. Sirbu and J. Tygar. NetBill: an Internet commerce system optimized for network-delivered services. *IEEE Personal Communications*, 2(4):34-39, August 1995.

[Sul00] B. Sullivan. Scam artist copies PayPal web site. MSNBC.com, Redmond, Washington, July 21st 2000.

[Sun97] Sun Microsystems. JavaCard 2.0 application programming interfaces. Revision 1.0, Palo Alto, California, October 1997.

[Sun98a] Sun Microsystems. Java remote method invocation – distributed computing for Java. Sun Microsystems White Paper, Palo Alto, California, 1998.

[Sun98b] Sun Microsystems. JavaCard applet developer's guide. Revision 1.12, Palo Alto, California, August 1998.

[Sun99] Sun Microsystems. JavaCard 2.1 platform specifications. Palo Alto, California, June 1999.

[Sun00] Sun Microsystems. Mobile information device profile (JSR-37), version 1.0, September 2000.

[SUSI99] ACTS Project AC320 SUSIE. Premium IP services. SUSIE Deliverable 4. April 1999.

[SUSI00] ACTS Project AC320 SUSIE. Project recommendations. SUSIE Deliverable 7. February 2000.

[SV93] M. Shand and J. Vuillemin. Fast implementations of RSA cryptography. In *Proceedings of the 11th IEEE Symposium on Computer Arithmetic*, pp.252-59, Windsor, Canada, June 1993.

[SV97] J. Stern and S. Vaudenay. SVP: a flexible micropayment scheme. In *Financial Cryptography '97 Proceedings*, pp. 161-71, Lecture Notes in Computer Science vol. 1318. Springer-Verlag, Berlin, 1997.

[SW00] B. Schneier and D. Whiting. A performance comparison of the five AES finalists. In *Proceedings of the 3rd AES Candidate Conference*, New York, April 2000.

[Sys99] Systemics Ltd. Cryptix for Java, version 3.1.1. October 1999. Available from http://www.cryptix.org/

[Sza96] N. Szabo. The mental accounting barrier to micropayments. 1996. Available from http://www.best.com/~szabo/

[Tan95] L. Tang. A set of protocols for micropayments in distributed systems. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, pp.107-15, New York, July 1995.

[Telc99] Telcordia Technologies GR-1100-CORE: Billing automatic message accounting format (BAF) generic requirements, Telcordia Technologies, December 1999.

[Tew95] H. Tewari. A flexible electronic payment system – FlexiCheck, M.Sc thesis, Trinity College Dublin, October 1995.

[TL96] L. Tang and S. Low. Chrg-http: A tool for micropayments on the World Wide Web. In *Proceedings of the 6th USENIX UNIX Security Symposium*, pp.123-29, San Jose, California, July 1996.

[TOP98] H. Tewari, D. O'Mahony, and M. Peirce. Reusable off-line electronic cash using secret splitting. Dept. of Computer Science Technical Report TCD-CS-1998-27, Trinity College Dublin, December 1998.

[Tri00] Trintech Group Plc, Annual Report Form 20-F, US Securites and Exchange Commission, January 2000.

[Tsi97] Y. Tsiounis. Efficient electronic cash: new notions and techniques. Ph.D thesis, Northeastern University, Boston, Massachusetts, June 1997.

[Tyg96] J. Tygar. Atomicity in electronic commerce. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, pp. 8-26, Philadelphia, May 1996.

[UMTS98] UMTS Forum Report #2: The path towards UMTS – technologies for the information society. London, October 1998.

[USEC99] ACTS Project AC336 USECA. Intermediate report on a PKI architecture for UMTS. UMTS Security Architecture (USECA) Deliverable 9, November 1999.

[VCFG+00a] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. AAA authorization appliation examples. IETF Internet Draft, March 2000.

[VCFG+00b] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. AAA authorization framework. IETF Internet Draft, March 2000.

[VvT97] E. Verheul and H. van Tilborg. Binding Elgamal: a fraud-detectable alternative to key-escrow proposals. In *Advances in Cryptology – EUROCRYPT '97 Proceedings*, pp. 119-33, Lecture Notes in Computer Science vol. 1233. Springer-Verlag, Berlin, 1997.

[VV98] J. Vandewalle and E. Vetillard. Developing smart card-based application using JavaCard. In *Proceedings of the 3rd Smart Card Research and Advanced Application Conference (CARDIS'98)*, Louvain-la-Neuve, Belgium, September 1998.

[Wal99a] B. Walke. *Mobile Radio Networks*. John Wiley & Sons, Chichester, USA, 1999.

[Wal99b] P. Wallich. How to steal millions in chump change. *Scientific American*, 281(2):21, August 1999.

[WAP98] WAP Forum. Wireless application protocol architecture specification. April 1998.

[WAP99a] WAP Forum. Wireless application protocol (WAP) wireless transport layer security (WTLS) specification. November 1999.

[WAP99b] WAP Forum. Wireless application protocol (WAP) identity module (WIM) specification. November 1999.

[WAP00] WAP Forum. WAP Certificate and CRL Profiles. March 2000.

[WCSW+99] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. TheVersaKey framework: versatile group key management. *IEEE Journal on Selected Areas in Communications*, 17(9): 1614-31, September 1999.

[Whe96a] D. Wheeler. Transactions using bets. In *Proceedings of the 4th Security Protocols International Workshop (Security Protocols)*, pp. 89-92, Lecture Notes in Computer Science vol. 1189. Springer-Verlag, Berlin, 1996.

[Whe96b] D. Wheeler. MicroMint extensions. Computer Laboratory, University of Cambridge, UK, November 1996.

[Whi97] P. White. RSVP and integrated services in the Internet: a tutorial. *IEEE Communications Magazine*, 35(5):100-6, May 1997.

[Wie98] M. Wiener. Performance comparison of public-key cryptosystems. *RSA Laboratories' CryptoBytes*, 4(1):1-5, Summer 1998.

[Wil80] H. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, 26(6):726-9, November 1980.

[Wil00] S. Williams. IrDA: past, present and future. *IEEE Personal Communications*, 7(1):11-9, February 2000.

[WM89] R. Walpole and R. Myers. *Probability and statistics for engineers and scientists*, 4th Edition. Macmillan, New York, 1989.

[WPRW+99] D. Wilcox, L. Pierson, P. Robertson, E. Witzke, and K. Gass. A DES ASIC suitable for network encryption at 10 Gbps and beyond. In *Proceedings of the 1st International*

Workshop on Cryptographic Hardware and Embedded Systems, pp. 37-48, Lecture Notes in Computer Science vol. 1717. Springer-Verlag, Berlin, 1999.

[WRCB+00] H. Wang, B. Raman, C. Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J. Shih, L. Subramanian, B. Zhao, A. Joseph, and R. Katz. ICEBERG: an Internet core network architecture for integrated communications. *IEEE Personal Communications*, 7(4):10-9, August 2000.

[WS00] X. Wang and H. Schulzrinne. An integrated resource negotiation, pricing, and QoS adaptation framework for multimedia applications. To appear in *IEEE Journal on Selected Areas in Communications*, vol. 18, 2000.

[WW98] W. Wang and C. Wu. A protocol for billing mobile network access devices operating in foreign networks. In *Proceedings of the 7th IEEE International Workshop on Enabling Technologies: Infrastucture for Collaborative Enterprises (WETICE '98)*, pp.262-8, Palo Alto, California, July 1998.

[WWW98] World Wide Web Consortium. Extensible markup languague (XML) 1.0. W3C Recommendation, Massachusetts Institute of Technology, Massachusetts, February 1998.

[WWW99] World Wide Web Consortium. Common markup for micropayment per-fee-links. World Wide Web Consortium, W3C Working Draft, Massachusetts Institute of Technology, Massachusetts, August 1999.

[WZ99] K. Wrona and G. Zavagli. Adaptation of the SET protocol to mobile networks and to the wireless application protocol. In *Proceedings of European Wireless '99*, pp. 193-8, Munich, October 1999.

[XYZZ99] S. Xu, M. Yung, G. Zhang, and H. Zhu. Money conservation via atomicity in fair off-line e-cash. In *Proceedings of Information Security Workshop (ISW'99, Kuala Lumpur, Malaysia)*, pp.14-31, Lecture Notes in Computer Science vol. 1729. Springer-Verlag, Berlin, November 1999.

[Yac97] Y. Yacobi. On the continuum between on-line and off-line e-cash systems – I. In *Financial Cryptography '97 Proceedings*, pp. 193-201, Lecture Notes in Computer Science vol. 1318. Springer-Verlag, Berlin, 1997.

[Yac99] Y. Yacobi. Risk management for e-cash systems with partial real-time audit. In *Financial Cryptography '99 Proceedings*, pp.62-71, Lecture Notes in Computer Science vol. 1648. Springer-Verlag, Berlin, 1999.

[YHH99] S. Yen, L. Ho, and C. Huang. Internet micropayment based on unbalanced one-way binary tree. In *Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99)*, pp.155-62, Hong Kong, July 1999.

[YK98] S. Yen and P. Ku. Improved micro-payment system. In *Proceedings of the 8th National Conference on Information Security*, Taiwan, May 1998.

[YLH99] S. Yen, C. Lee, and L. Ho. PayFair: a prepaid Internet micropayment scheme promising customer fairness. In *Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99)*, pp.213-21, Hong Kong, July 1999.

[Yuv79] G. Yuval. How to swindle Rabin. Cryptologia 3(3):187-9, July 1979.

[ZC99] G. Zorn and P. Calhoun. Limiting fraud in roaming. IETF Internet Draft, September 1999.

[Zha98] K. Zhang. Efficient protocols for signing routing messages. In *Proceedings of the 1998 Internet Society (ISOC) Symposium on Network and Distributed System Security*, San Diego, California, March 1998.

[ZL98] J. Zhou and K. Lam. Undeniable billing in mobile communication. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98)*, pp.284-90, Dallas, Texas, October 1998.

[ZL99] J. Zhou and K. Lam. A secure pay-per view scheme for Web-based video service. In *Proceedings of the 2nd International Workshop on Practice and Theory in Public Key Cryptography (PKC'99)*, Lecture Notes in Computer Science vol. 1560. Springer-Verlag, Berlin, 1999.

[ZPS92] Y. Zheng, J. Pieprzyk, and J. Seberry. HAVAL – a one-way hashing algorithm with variable length of output. In *Advances in Cryptology – AUSCRYPT '92 Proceedings*, pp. 83-104, Lecture Notes in Computer Science vol. 718. Springer-Verlag, Berlin, 1992.