Mudra: An Electronic Payment Scheme for Networks

Vishv Malhotra
Department of EE & CS
University of Tasmania,
GPO Box 252-65 Hobart 7001,
Australia
vishv.malhotra@utas.edu.au

Bala Srinivasan
Department of Computer Technology
Monash University
Caulfield 3145,
Australia
srini@ct.monash.edu.au

Abstract *Revenue collection services are deemed essential for attracting quality services on the networks. Generally these collections will be of small amounts but will be frequent. A water-tight authorisation check is not cost-effective in such an environment. This paper introduces an inexpensive, robust and fast scheme for authorising the credit usage. The scheme is especially suited to small credit brokers as it does not involve any special hardware.*

INTRODUCTION

In his recent article '*Digital Libraries, Value, and Productivity*', Wiederhold [Wie95] declares "The services discussed in this article will require fees from customers. The fee for a unit of service must be fairly low, since many services will be needed to support decisions". Indeed, efficient systems for small amounts, high frequency revenue transactions are of paramount importance in attracting quality services on the networks. Credit and debit cards are widely used in the real world for payment of goods and services. Their usage is authenticated by physically producing a card and a signature at the point of its use. However, the use of these cards over a network is fraught with many difficulties. Much of the difficulty results from the ability of the computers to duplicate information in a way that does not distinguish the original from a copy. This makes the use of simple account numbers as authorisation on a network unacceptable.

In this paper we develop a simple scheme called Mudra[1] that a small credit broker can adopt to provide a revenue transaction service over the networks for a community of consumers and service providers. The salient features of the scheme are its high-security, off-line operation and no need for any special hardware.

For the purpose of this presentation we define a number of actors on a network. There are a number of *service providers* providing diverse services on the

[1] Mudra is a Sanskrit word meaning coin(s).

networks. The *consumers* use the services provided by these providers. We assume the existence of a special class of service providers called *creditors*. A creditor undertakes to pay the service providers on behalf of the consumers. The consumers register with a creditor and are assigned account numbers. They must nominate sets of addresses at which they will access the services. A consumer seeking a service makes available to the service provider an account number and an address at which the service is required. We shall use the term *capability* to denote a pair of an account number and an address. The service provider verifies that the account is valid for the specified address before delivering any service to the address. As an acknowledgment for the service received, the consumer hands in an *acktoken* to the service provider. Finally, but not the least interesting actor on the network is a *cheat*. A cheat tries to have a service delivered at the expense of other consumers. The aim of this paper is to provide a revenue collection mechanism that is efficient and frustrates the cheats.

An obvious solution is to provide a complete list of account number-address pairs (capabilities) to all service providers. Each request for a service is accompanied by a capability. The service is delivered only if it is demanded at an address (site) that the capability permits. The restricted nature of capabilities frustrates the cheats from gaining unauthorised access to the services. A typical consumer accesses network services from only a few sites. The added vulnerability of the accounts that are valid everywhere is thus avoided.

The amount of data involved, however, makes the solution impractical. It entails a massive transfer of data to each service provider. It also requires a search over a large database to check that the consumer has a valid capability. Further, an enterprising service provider may find it too tempting and distribute unwanted advertisements to the addresses.

In this paper we describe a practical implementation of the above idea. The implementation does trade-off some security for reduction in the amount of data needed to verify the capabilities. Some additional features of the implementation are:

1. The security data is distributed in an encoded form. The coded data cannot be used to regenerate the original data.

2. The amount of data needed by a service provider depends on the level of risk involved. Larger amount of data provides higher security.

3. A cheat who succeeds in obtaining an unauthorised service from a provider does not automatically gain access to other services as well.

4. The basic idea outlined above is augmented to provide safeguards against cheating by a consumer or a service provider.

5. The data structures needed in the implementation can easily grow or shrink as the consumer population changes.

The item (3) above is unique to this scheme and has not been reported in the other electronic payment schemes. A comprehensive electronic payment scheme must counter unauthorised access to the services at three levels:

1. Prevent a cheat from gaining access.

2. Detect successful as well as unsuccessful attempts by cheats to gain accesses.

3. Limit the access of a successful cheat to a small set of services and for a short period.

A scheme focused exclusively on only one aspect is vulnerable and subject to security risks. The risk is especially serious if a cheat can attack the scheme off-line or can remain anonymous by simply spreading the attempts to cheat over several service providers.

We briefly survey the related works in Section 2 and bring out the issues of interest in electronic money transaction systems. In Section 3 we describe a bit hash algorithm and associated data structure for storing and verifying the capability base of the system. We analyse the risk inherent in the algorithm and show that it is low. An implementation of the algorithm on a network is suggested in section 4. We conclude the paper in section 5 with some remarks.

RELATED WORK

It is helpful here to characterise the nature of the services that the payment scheme described in this paper targets. The traditional services — services in the physical world — available from the remote sites have involved delivery of the goods and packets at a specified postal address or geographic location. The delivery could be easily coupled with the payment arrangements. At the very least, the delivery location provides assurances regarding the identity of the consumer. The nature of the services provided over the networks is radically different. These are services delivered from automated sources involving little human intervention. The traditional services have physical constraints on the number of services that can be delivered. Thus, delivering a service to one consumer excludes another consumer from receiving the same service. For example, a tube of toothpaste taken by a consumer from a supermarket shelf

reduces the number on the shelf by one. Likewise, a doctor examining a patient cannot serve another patient in the same time period. As opposed to these, a consumer playing a game over a network does not reduce[2] the number of games that can be played later. Thanks to multi-programming, a game can concurrently be delivered to many consumers. A perfect revenue collection scheme may not be essential in this environment. A collection service that collects most of the payments is acceptable; especially if it is efficient, flexible and not too leaky.

A number of proposals and products are available for effecting electronic transactions on the networks. Virtually everyone of them aims at providing high level of security with no leakage. A typical method uses encryption [RSA78, DiH76] to exchange information in a way that enables verification without revealing all details. For example, CyberCash (`http:// www.cybercash.com`) makes available to the consumers an RSA-based encryption tool. The consumer transmits the encrypted account information to the service provider. The service provider adds the transaction details to this information and sends it to the creditor for verification before delivering a service. NetCash (`http:/ www.netbank.com/~netcash/`) makes available small denomination coins that carry a sparse serial number. The idea is to ascertain that the number matches the claimed denomination. Double use of the coins is prevented through a close supervision of the transactions by a currency server. Mondex (`http://www.mondex.com /mondex /home.htm`) uses a Smartcard to create an electronic purse that can be used to pay real money over the networks. LETSystem (`http:// www.u-net.com/ gmlets/`) allows the consumers and service providers to execute financial transaction through instructions to the creditor. On instruction from the consumer the specified amount is transferred from the consumer's account to the service provider's account. DigiCash (`http:// www.digicash.com/`) uses patented products to provide privacy (anonymity) without compromising the accountability on the part of its users [Cha92]. With suitable arrangements, it is possible to conduct off-line transactions under this system. The ESPRIT project CAFE [BBC94] also provides a system for financial transactions using custom-build hardware. Millicents from Digital Equipment Corporation (DEC) (`http:// www.millicent.digital.com/`) provides a software script to handle the monetary part of the transactions. This big brother approach is not uncommon among the products from other card providers and banking groups.

As can be seen, a typical transaction is completed with the aid of a server (creditor) trusted by both parties. These systems adequately address the needs of transactions involving larger amounts, but are too expensive for the small amount transactions that we expect will be common on the networks. The total

[2] Here we are concerned with physical changes and ignoring issues like licensing.

cost of a complete transaction is double the cost of the provider's service if the creditor's service costs about the same as that of the service provider. The centralised nature of the clearing is also a bottleneck for many of the above-mentioned systems.

More recently, some proposals for micro-payment schemes have been made which do not require an on-line creditor to arbitrate on each payment. *Electronic Lottery Tickets and Micropayments* proposed by Rivest (`http://theory.lcs.mit.edu/~rivest/ publications.html`) suggests a scheme where a creditor sells lottery tickets to the consumers which they use for purchasing services. A service provider only receives payments for winning tickets. Thus, the scheme overcomes the problem of excessively large number of uneconomically sized monetary transactions between the creditor and service providers. Further, the scheme relies on the law of large numbers to assert that each service provider will receive a fair payment and each consumer will pay their fair share. The scheme clearly suffers from its random nature which may place many service providers in severe and unnecessary cash flow problems. It also fails to address the issue of collusion between the parties.

Rivest and Shamir [RiS96] have recently proposed two other payment schemes, PayWord and MicroMint. A transaction using PayWord requires consumers to create a chain of related pay-words using a hash function. A service provider can verify that the pay-words being given are from the same sequence by validating each payment (pay-word) against the previous pay-word received from the same consumer. It is, however, not clear how the scheme prevents a cheat (or service provider) from creating counterfeit pay-words. This in turn opens the possibility of a consumer repudiating the payment. The security of the scheme is simply based on the fact that a creditor will exclude from the scheme a consumer or a service provider who is involved in excessive number of disputes. It also requires that the service provider present the PayWord to the creditor no later than a day after receiving them. MicroMint can easily create a collection of easily verifiable coins. Though this is an improvement over NetCash scheme discussed earlier, it continues to rely on non-anonymous transactions and “vendors’ honest cooperation” to prevent consumers from double spending. Further successful forging of the coins is possible. The proposers of the scheme suggest that this can be countered by creating new coins at the start of each time period or even recalling all coins by declaring the period over.

The discipline is still evolving, new proposals and protocols for payment schemes are constantly emerging. An excellent collection of links to network payment systems from established banks, credit card companies and just starting groups is available at `http://ganges.cs.tcd.ie/ mepeirce`

`/Project/ oninternet.html`. Another rich web site devoted to the issues is `http:// robotics.stanford.edu/ users/ ketchpel/ ecash.html`. It is evident from the proposals surveyed in this section that money transactions on the networks have risks that are very different in nature from those in the physical world. Security has many facets each with its cost-to-benefit pay-offs. It is, however, clear that none of the proposed payment schemes counter the cheats at all three levels identified in the introduction.

Security is not the only requirement of a payment scheme. A paper describing NetCash [MeN93] introduces a number of useful requirements for electronic currency. *Security* against forging, *anonymity* to protect privacy, *scalability* that allows the system to avoid single server bottleneck, *off-line operation* to conduct transactions without creditors intervention, and *independence from hardware* are some of the desired properties of the electronic money.

A transaction that a consumer can repudiate is clearly insecure. Chaum [Cha85] provides an insight into the security and privacy issues. We need a digital signature algorithm to create records of transactions that cannot be denied by the consumers. The basic idea of a digital signature is to have a pair of keys. One of the keys is a secret encryption key known only to the creator of the signed message. This key is used to encrypt (sign) the messages. The other key is a public decryption key that can be used to read signed (decrypt) messages[3]. There are algorithms available, see [Sim92] for a comprehensive collection of articles, that generate the pairs of encryption and decryption keys in which it is very difficult (that is, it costs more than the potential benefits) to construct encryption keys from the publicly available decryption keys. Thus, anyone with access to the decryption key can read the message but cannot forge one.

In remainder of this paper we present a scheme that provides a method for completing transactions without needing an on-line mediator.

DATA STRUCTURE FOR REPRESENTING CAPABILITIES

In this section we introduce algorithms and data structures for efficiently encoding and verifying capabilities. The data structure is called multiple bit hashing tables [MSK95]. As the name suggests, this technique uses multiple hash tables. The key idea is to use a set of independent hash functions to encode capabilities onto a set of tables of bits. A hash function maps its argument capability onto an index over the table; that is, an integer in the range 1 through

[3] In this paper, we use the terms *encrypt* and *sign* interchangeably. Similar remark applies to the terms *decrypt* and *read a signed* message.

the size of the table. There is a table for each hash function. Initially, all bits in each table are cleared. The bits in a table are set by using its associated hash function. The functions are applied to a capability to generate indices – one index for each table. The corresponding bits in the tables are then set to 1.

During verification of a capability, a similar procedure is followed. For each table, an index is obtained by applying the hash function to the capability. The bit at the indicated position is examined. This procedure is continued to the next table if the bit is set. If the examined bits in all tables are found to be set then the capability is declared verified; the service provider may provide the service. If, however, a clear bit is detected during the verification, the capability is invalid and the service is denied.

In order to illustrate the idea we present an example where we try to set three capabilities K = {<vmm,eecs.utas.edu.au>, <srini, ct.monash.edu.au>, <vmm,ct.monash.edu.au>} in an empty data structure. We assume a total available space of 30 bits. The total space is divided into 3 hash tables T_1, T_2, and T_3 of size 10 bits each. We also assume 3 hypothetical hash functions h1(), h2() and h3() corresponding to these tables. The choice of the number of hash tables and the table sizes is an important factor as it determines the performance of the data structure. We discuss this issue later in this section. To start with, all the bits in the tables are clear. In order to insert the first capability, <vmm, eecs.utas.edu.au> in the data structure, we apply the hash functions h1(), h2() and h3() to it. Let us assume that after applying the hash functions we get the indices 1, 6 and 2 respectively. Therefore the 1^{st} bit of T_1, 6^{th} bit of T_2 and 2^{nd} bit of T_3 are set to 1. Figure 1 shows the hash tables after inserting the capability. A similar procedure is followed to encode the other capabilities in the hash tables.

Let us assume that after inserting all the capabilities, the hash tables appear as shown in Figure 2. In order to verify whether the capability <vmm, eecs.utas.edu.au> is present in the data structure or not, we apply hash functions h1(), h2() and h3() and get the resulting indices 1, 6 and 2 respectively. Next we examine the 1^{st} bit of T_1, 6^{th} bit of T_2 and 2^{nd} bit of T_3. Since all these bits are set, we conclude that the capability is present in the data structure.

The use of a set of single bits to represent a capability makes this technique very economical in terms of space. During verification, there are no string comparisons. This makes the process fast compared to the techniques which involve string comparisons.

A problem arises when a capability that is not present in the data structure (say, <cheat,pc.jail.edu>) returns the indices 3, 6 and 2 after applying the hash functions. In such a case, one would examine the corresponding bits and conclude that the capability exists, which is not true. This results in a *false verification* and constitutes a risk that is inherited in the system of providing service to a cheat. In what follows we show that the risk can be set to any stipulated small level in a proper design of a multiple bit hashing structure.

Properties of multiple bit hashing

In this section, we analyse and predict the risk that a creditor inherits by using a system of multiple bit hashing tables. For this purpose, we will assume that a hash function maps capabilities uniformly over the locations in the table. We shall derive an expression to determine the number of bits that are expected to be set as capabilities are mapped into a table of `n` bits. This result is then used to derive an expression for determining the risk in a system of `t` independent bit hashing tables.

	T1	T2	T3
1	1	0	0
2	0	0	1
3	0	0	0
4	0	0	0
5	0	0	0
6	0	1	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

Figure 1: Hash Tables after inserting the capability <vmm,eecs.utas.edu.au>

	T1	T2	T3
1	1	0	0
2	0	1	1
3	1	0	0
4	0	0	0
5	0	0	1
6	0	1	0
7	0	0	0
8	1	0	0
9	0	0	0
10	0	1	0

Figure 2: Hash Tables after inserting all (three) capabilities

Let

w be the number of capabilities,

t be the number of tables used,

n be the number of bits in a single hash table,

N $= n \times t$, be the total number of bits used for the bit hash tables. This is also the total size of the data structure

b $= N / w$, be the number of bits of memory available for each capability.

Number of bits set in a table

Let $f(i)$ be the number of bits set in a table of size n after i capabilities have been entered. It is easy to see that the following recurrence relation for $f(i)$ holds:

$f(0) = 0$

$f(i+1) =$ bits set by first i capabilities
+ probability that $i+1$th capability sets a new bit

$$= f(i) + (1 - f(i)/n)$$

$$= 1 + f(i)(1 - 1/n) \quad (1)$$

The recurrence relation can be solved to get the following closed form solution:

$$f(i) = n\left(1 - \left(1 - \frac{1}{n}\right)^{i}\right) \quad (2)$$

By setting $i = w$, we get the number of bits set when all capabilities have been encoded into the table. For a typical (large) value of w the expression (2) approximates to

$$f(w) \cong n\left(1 - e^{-\frac{w}{n}}\right) \quad (3)$$

Creditor's Risk

The probability that an invalid capability hashes onto a set bit in a table is determined by the ratio of set bits to the total number of bits in the table, that is, $f(w)/n$. A service provider will wrongly deliver a service if the capability hashes onto a set bit in each of the t tables. Let $error(t,b)$ denote the probability of this event. Clearly,

$$error(t, N/w) = \left(\frac{f(w)}{n}\right)^{t}$$

$$= \left(1 - \left(1 - \frac{t}{N}\right)^{w}\right)^{t} \quad (4)$$

$$\cong \left(1 - e^{-\frac{t\,w}{N}}\right)^{t} \quad (\text{as } n = N/t)$$

$$error(t,b) = \left(1 - e^{-\frac{t}{b}}\right)^{t} \quad (5)$$

It can be verified, using the standard methods of calculus, that this expression assumes its minimum value when

$$t = t_{optimal} = b \log_e 2 \quad (6)$$

It follows from this result that at the minimum risk, the organisation of each table has $w / \log_e 2$ bits. This corresponds to $f(w)/n = 1/2$. And, hence

$$error(t_{optimal}, b) = 2^{-b \log_e 2} \quad (7)$$

The above expression denotes the creditors risk, so we write

$$risk(b) = 2^{-b \log_e 2} \quad (8)$$

Bits-per-capability vs Creditors Risk

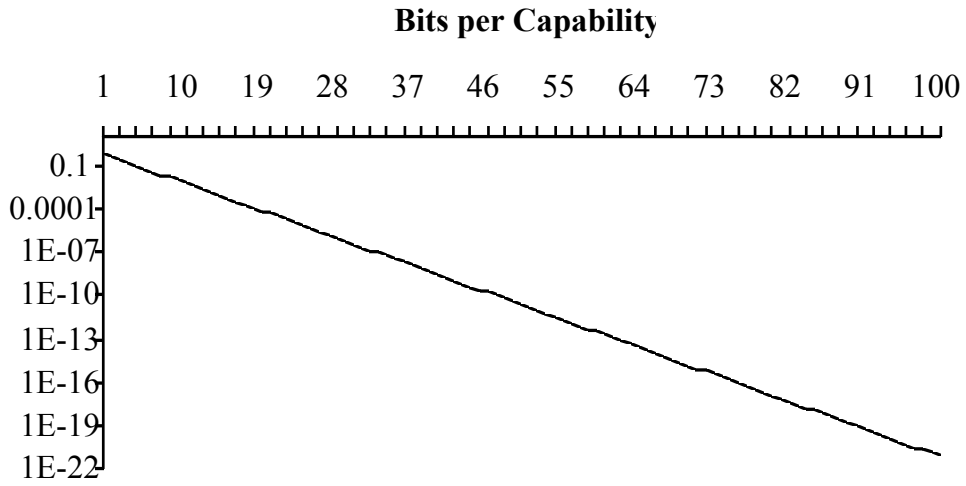

Figure 3: Graph depicting the creditor's risk as a function of bits used for encoding the capabilities.

The graph in Figure 3 shows the creditor's risk against the number of bits used to encode a capability. As can be seen from the graph, 30 bits per capability gives less than 1 chance in a million for a cheat to gain unauthorised access to a service.

MUDRA: AN ELECTRONIC PAYMENT SERVICES

The implementation of the proposed electronic payment service, *Mudra*, is woven around the capabilities. Each request for a service must be accompanied

by a capability. A service provider must verify the capability before accepting it. Thus, capability verification centres are fundamental to the scheme as the mechanism for verifying capabilities as being valid.

Capability Verification Centres

The bit hashing tables provide a practical method for capability verification on the networks. The creditor creates a large repository of hash functions. These functions are then used to fill bit-hashing tables with the encoding of current capabilities. Sets of these tables together with the associated hash functions are installed at the capability verification centres. Major service providers may have centres operating on their sites. The smaller ones may decide to pool their resources and use a common capability verification centre.

The sets of tables installed at various centres need not be the same. Nor is there any need to place the same number of tables at each centre. The number of tables can vary between the centres and over time depending on the level of the required security.

The tables kept at various centres, however, are not fixed. The tables will change as new consumers register with the creditor and as existing consumers leave the creditor. These changes will need the tables installed at capability verification centres to be updated regularly. It is easy to install new capabilities in the bit-hash tables. The deletion of the capabilities, on the other hand, is not simple.

To add new capabilities the creditor needs to set new bits in the tables already installed at the verification centres. A new consumer will be able to receive a service from a service provider when all affected tables at the associated capability verification centre have been updated. As more centres get updated, the new consumers gain access to a wider range of services on the networks.

A similar delay will be experienced before all capabilities of a resigning consumer are purged from the system. To this end, the creditor regularly replaces old tables from the capability verification centres by new tables containing the current capability encoding. Fortunately, not all centres need to be changed simultaneously. Nor is it necessary to replace all tables at a centre simultaneously. The tables can be replaced one at a time over an extended period of time. As one or more tables are replaced from a centre, the capability encoding of the resigned consumers cease to exist at the centre. To reduce the risk of an undetected cheat continuing to receive service from a centre it is useful for the replacement table to be based on a different hash function. This is also an opportunity to vary the size of the table in response to changes in the consumer population.

Transaction Acknowledgment

The capabilities provide security against cheats but suffers from some annoying limitations. A service provider may double claim for a service. On the other hand, a consumer may deny having received a service. For example, suppose a service provider claims that a consumer used a capability to receive two services A and B. The consumer may insist that only one service was received. It is not possible to resolve the conflict without additional information. Another example of a difficulty is if a service provider makes available a consumer's capability to another provider. The latter provider may claim payment for a service without having provided any service. In rest of this section we focus on eliminating cheating by the consumers and service providers. This is achieved by augmenting the basic system with a procedure for acknowledging the services. To be useful each service must be acknowledged by a non-forgeable but distinct acknowledgment.

The creditor creates a large number of signed acktokens and associates with each acktoken a pair of digital signature keys: an encryption key to sign the acknowledgment and a decryption key to read the signed acknowledgment. The decryption key is then signed by the creditor using her personal signature key. This is done to prevent the consumers from creating unauthorised keys. Each service provider knows the creditor's decryption key. They can use this decryption key to be sure that the decryption key being supplied by the consumer was created by the creditor. Each consumer receives a random set of these acktokens together with the signature key pairs. The use of signature keys is explained later.

An acktoken is given by a consumer to the service provider as a part of the acknowledgment for the service. It is crucial to ensure that a service provider accepts an acktoken from a consumer only once. The fact that a service provider holds an acktoken that only the consumer (besides the creditor) knows is strong evidence that a service was provided. In the next couple of paragraphs, we present an algorithm for efficiently checking the uniqueness of the acktokens. Besides efficiency, the issue of security is also important – a consumer may not like to reveal an acktoken to the service provider before the service has been provided.

The bit-hashing tables are once again a useful data-structure for ensuring the uniqueness of the acktokens received by a service provider from the consumers. The service provider may use a set of hash functions to convert consumer account-acktoken pairs into indices over the bit-hashing tables. These indices are recorded in an appropriate set of tables by the service provider. A duplicate acktoken is indicated if all indices generated by the hashing functions for the pair map onto the set bits in the tables.

If a standard set of hash functions are used for converting consumer's account-acktoken pair into indices then the uniqueness check can be made without revealing the acktoken to the service provider. The consumer applies the hash functions and conveys the indices to the service provider. The service provider checks the indices against his data structure to either indicate that the acktoken is acceptable or to suggest that the consumer selects a different acktoken. The acktoken is given to the provider only after the service has been provided: the acktoken is placed by the consumer in a record together with the decryption key and other transaction details. The record is signed (encrypted) by the consumer before transferring it to the service provider. To ensure that the consumers use the correct signature-key pair with an acktoken, all acktokens are paired with their decryption keys by the creditor and signed before they are given to the consumers. The service provider receiving an acknowledgment record must decrypt the record and ensure that it is valid.

The anonymity of the transactions is preserved as long as there is no dispute between the consumer and the service provider. At the end of each accounting cycle, the service provider requests for a payment of the fees for all services provided to an account. If the consumer and the service provider kept their records carefully, the creditor will easily be able to verify the consolidated fee from the consumer. There is no need for the creditor to know the transaction details. However, the creditor will need to call for the details if the parties disagree.

A complete interaction between a consumer and a service provider is shown in Figure 4. The basic idea of the protocol is to first establish that a capability exists. This is done in step marked 1 in the figure. The consumer transmits the account number and address where the service is required. An eavesdropper will not benefit as the capability check will fail if the service is requested for a different address. The consumer and the service provider then negotiate an acktoken that the consumer will use to acknowledge the service. Steps 2 and 3 represent this negotiation. At the end of step 3, the provider has verified that a valid capability exists and the parties have chosen an acktoken to be used for acknowledging the service. However, the provider needs to confirm that the consumer owns the acktoken. Only an owner of an acktoken has the encryption key associated with it. Steps 4, 5 and 6 provides this confirmation. The consumer sends a decryption key signed by the creditor. The service provider responds by sending a random number challenge in step 5. The consumer encrypts the number to convince the service provider that he has the encryption key. The interaction secures the capability system against a cheat who is able to spoof [Bel89] an address. The service provider agrees to provide a service in step 7. The request for a service is made by the consumer in step 8. The request is signed by the consumer and may carry some additional details. The idea is to

create an evidence for a request in case the consumer fails to transmit an acknowledgment after the service. The last two steps 10 and 11, show the exchange of an acknowledgment between the consumer and the service provider on completion of the service.

The algorithm provides a method for completing financial transactions on the networks between two parties – a typical transaction is expected to conclude successfully. However, the transactions may not complete successfully for four reasons:

- a system failure,
- a dishonest consumer transmitted incorrect indices in step 2,
- a dishonest consumer failing to execute step 10 after receiving a service, or
- a dishonest service provider failing to provide the requested service in step 9.

The records of interactions preceding a failure provide information for identifying the interacting parties. In this sense the present scheme is superior to other reported in the literature which fail to even establish if the two parties were involved in an interaction or one is just alleging that the other cheated. However, it may not always be possible to determine the erring party. A consumer or a service provider who is frequently involved in disputed transactions loses goodwill and eventually would be excluded by the creditor(s).To limit their exposure to dishonest consumers a provider of an expensive service may consider payments on per-stage basis.

CONCLUSION

The conventional approaches to revenue collection are based on encryption and passwords. The lynch pin of these approaches is the secrecy that makes it difficult for a cheat to guess the keys. However, once a cheat has guessed a key, the compromised system is open to exploitation. The identity of the cheat is often very difficult to trace. Even the fact that the system has been compromised may not be detected until other follow-up consequences emerge. The corrective actions may involve extensive overhaul of the system. These may require the system to be closed to normal business while an extensive time consuming overhaul is performed. A perfect, absolutely secure payment system is a worthy but an unrealistic goal. A pragmatic scheme must aim for high reliability and ability to function gracefully even when a part has been compromised. Currency systems in the physical world exhibit these characteristics and remains functional in spite of the counterfeit currency notes, defrauded cheques and misused credit cards. A three prong strategy was suggested in the introduction to manage the risks in an electronic payment scheme.

The method proposed in this paper exploits the fact that a typical consumer uses the network only from a small set of fixed addresses and thus makes the transactions secure. The nature of the capabilities prevent their use at addresses other than the nominated sites (addresses). A concerted attack by guessing capabilities needs to be for a single site and to the same capability verification centre. This is easily detected; thus, the service is provided in an environment where the identity of the interacting parties is well established. One is unlikely to indulge in a misdemeanour if the actions are easily traced back to them. The efficiency of the method depends on the risk level that the service demands. A lower risk level is attained by using more tables. The method is robust and does not need an immediate overhaul if a cheat breaks into the system. The cheat gets access to only a part of the system services and that too for a limited period. Even if a cheat remains undetected, the access to a service will eventually terminate as the tables are replaced and renewed at the capability verification centres. Indeed, the scheme thwarts a cheat's attempt to gain access at all three levels identified in the introduction.

Another strong point of the proposed scheme over the other schemes is its scalability and off-line operation. The transactions complete without any intervention from a central authority. This makes it a useful system that needs only small setting-up and running cost. The creditors are not required to keep their systems on-line around the clock. Nor does the scheme require any special custom-build hardware for its implementation.

ACKNOWLEDGMENTS

We wish to acknowledge and thank Drs. Charles Lakos and Cristina Cifuentes for many helpful comments. Two anonymous reviewers also provided valuable comments and criticism to improve the quality and currency of this work.

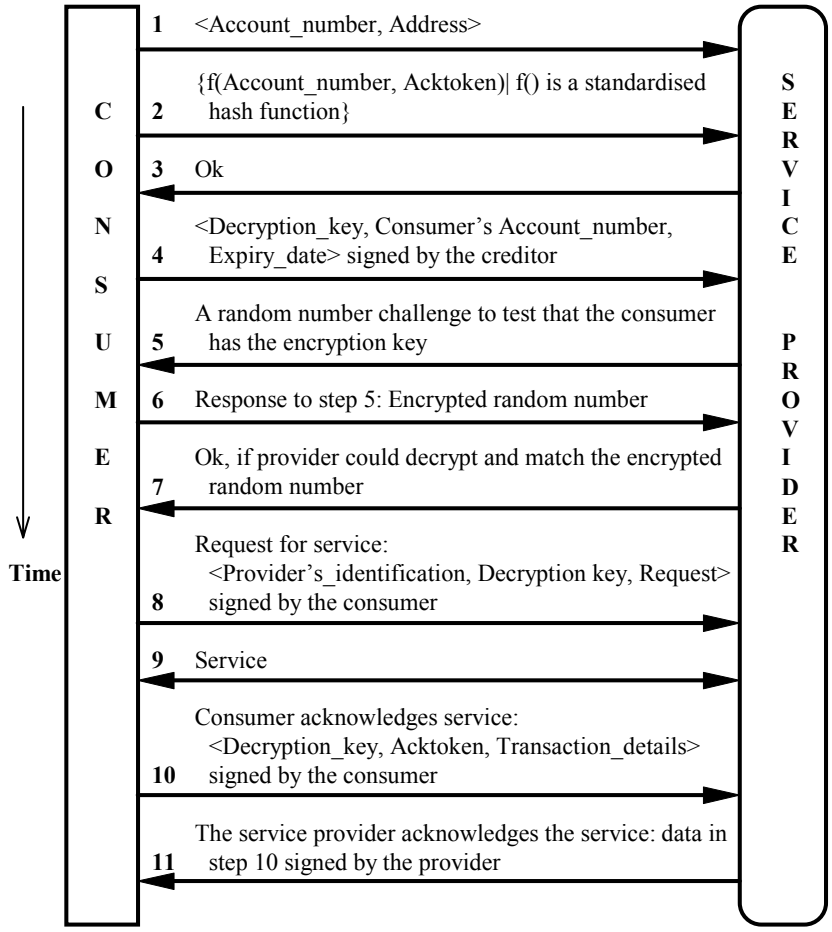

Figure 4: A complete transaction between a consumer and a service provider. The consumer starts by declaring a capability (Step 1). If the service provider can verify the capability, the consumer continues to provide enough further information to let the provider be sure that the acktoken will not be a repeat (Step 2). A certified decryption key is then sent to the provider (step 4) who in turn ensures that the consumer holds the encryption key (steps 5-7). The request for the service (step 8) is sent after the provider has verified that the consumer is genuine. After the service, the consumer (step 10) and the service provider (step 11) exchange signed (encrypted) acknowledgments for the service.

REFERENCES

[Bel89] Bellovin, S.M.(1989), Security Problems in the TCP/IP Protocol Suite, *Computer Communication Review,* 19(2), pp. 32-48, April.

[BBC94] Boly, Jean-Paul, A. Bosselaers, R. Cramer, R. Michelsen, S. Mjolsners, F. Muller et al (1994), *The ESPRIT Project CAFE – High Security Digital Payment Systems,* pp. 217-230, In: D. Gollmann (Ed.), *Computer Security – ESORICS 94*, LNCS 875, Springer-Verlag, Berlin.

[Cha85] Chaum, D. (1985), Security without Identification: Transaction Systems to Make Big Brother Obsolete, *Comm. ACM* 28(10), pp.1030-1044, Oct.

[Cha92] Chaum, D. (1992), Achieving Electronic Privacy, *Scientific American*, pp 96-101, Aug.

[DiH76] Diffie and M.E.Hellman (1976), New Directions in Cryptography, *IEEE Trans. on Inform. Theory,* vol. 22, pp. 644-654, Nov.

[MSK96] Malhotra V.M., B. Srinivasan and S. Kulkarni (1996), Storage Efficient Data Structure for Large Lookup Dictionaries, *Info. Processing Letters*, 58(4), pp. 201-206, May.

[MeN93] Medvinsky, G and B.B. Neuman (1993), NetCash: A design for practical electronic currency on the Internet, *Proc. of first ACM Conf. on Computer and Comm. Security*, November.

[RSA78] Rivest, R.L., A. Shamir and L.M. Ademan (1978), A., Method for Obtaining Digital Signatures and Public Key Cryptosystems, *Comm. ACM* 21(2), pp. 120-126.

[RiS96] Rivest R.L., A. Shamir (1996), PayWord and MicroMint: Two simple micropayment schemes, `http:// theory.lcs.mit.edu/ ~rivest/ publications.html`.

[Sim92] Simmons, G.J. (editor) (1992), *Contemporary Cryptology The Science of Information Integrity*, IEEE Press.

[Wie95] Wiederhold, G.(1995), Digital Libraries, value, and productivity, *Comm. ACM*, 38(7), pp. 85-96.