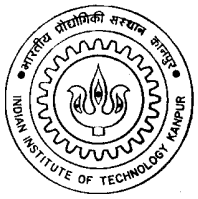
Secure and Efficient Micropayment Systems

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Sambasiva Rao.B



to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

February, 2001

Certificate

This is to certify that the work contained in the thesis entitled "*Secure and Efficient Micropayment Systems*", by *Sambasiva Rao.B*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

February, 2001

(Dr. Manindra Agarwal)
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

Abstract

Micropayments refer to low-cost financial transactions ranging from a few cents to a few dollars. A sizable portion of electronic commerce occurring in the Internet belongs to the category of micropayments. The transaction cost of micropayments should be kept as low as possible without sacrificing much on double spending, anonymity, non-repudiability and other required security properties.

We have developed a system for micropayments without the use of public key cryptography. The protocol is off-line and uses hash operations only. The off-line protocol allows the customer to buy coins at one broker and spend them with any vendor. The system can also be used as a payment mechanism for the transfer of larger payments.

Acknowledgements

I would like to take this opportunity to express my gratitude to my thesis supervisor Dr. Manindra Agarwal. He has helped me at every stage of this work with his innovative ideas and his deep knowledge of the subject, without which this thesis could not have been possible. He has always been very understanding, compassionate, encouraging and helpful to me.

I am thankful to all the faculty members of Computer Science and Engineering department, for helping me have a better understanding of various subjects of computer science. It was great to be in the company of Suresh, Kiran, Harikishan, Sunil, Baljit and many others. I would like to thank all my Telugu Association friends. I would like to thank Laxminarayana, Raghu, Sridhar, Balaji, Palankan, Vinod, Pandu, Sundar, Saritha and other friends for their constant support and inspiration

I would like to thank all my classmates of Mtech'99 batch for being supportive and making my stay at IIT Kanpur comfortable and memorable.

I am grateful to my parents, sisters and all my close relatives for their love and care.

Contents

List of Figures

Chapter 1

Introduction

The rapid growth of Internet has led to the appearance of thousands of different web sites, which are created to provide information for millions of people around the world. Producing and maintaining a quality web site takes a great deal of time, dedication and money. Hence the producers of many of these web sites wish to keep their site as a source of income by selling small chunks of information to users for a small amount. For example an online newspaper could decompose a newspaper into different sections like political, sports etc and sell different sections for a few cents. Other examples include individual stock quotes, delivering songs online etc. So the market for these low cost items is significantly high.

There are a number of different payment systems like SET, E-Cash etc. But in these payment systems the cost of processing the transaction is several cents, which will destroy any profit margin on these consumer items. So traditional protocols cannot be used for making these transactions. Hence the micropayment systems are designed whose primary objective is to reduce the cost of processing the transaction.

1.1 Principles and Parties

The various parties involved in transaction are

Customer

A Customer has an account with the broker, who will pay money to the vendors

after making sure that the customer has spent money with them. The customer is provided with a set of coins when he requests the broker and uses these coins in transactions.

Vendor

A Vendor is the one who is willing to provide services or information to customers in exchange of money. The vendor contacts brokers who assure him of the money that customers spent with these vendors.

Broker

Brokers tend to be large, well-known and reputable financial institutions(e.g Master Card,Visa, Banks,ISP's etc). They act as intermediaries between vendors and customers. They also resolve disputes between them and identify the cheating, if any. They charge a percentage of transactions either to Customers or to Vendors.

Adversary

The adversary is an abstraction of attacker of the system. There can be different kinds of adversaries. An eavesdropper, who just listens to messages and tries to learn secrets (e.g credit card numbers) or an active attacker who is introducing forged messages in an attempt to cause the system to misbehave. The most natural and strongest attack is a vendor who tries to cheat and get paid without authorization

An adversary is equipped with intelligence, money, hacking skills everything short of protected secret keys of other parties and also has the power to do a lot of computations.

1.2 Payment System Requirements

The various characteristics that a payment protocol should satisfy are

1.2.1 Anonymity

An anonymous protocol is one in which the identity of the customer should be hidden from the parties involved in the transaction. When discussing anonymity, it is not sufficient for a protocol merely to protect buyer's identity but it should also be impossible to identify or to compare purchases in such a way that whether the same person has made them.

1.2.2 Atomicity

Atomicity deals with how a protocol reacts when it is interrupted. A protocol is called atomic if a payment can only be completed in full or not completed. Interrupting the protocol should give the same result as never having started the protocol at all.

1.2.3 Non-repudiability

Non-repudiability deals with the question of whether a transaction can be denied by one of the participants. A non-repudiable protocol is one in which it can be proven that payment was made and accepted. i.e customer can prove that he or she made the payment and payee has received the payment. A strict version of non-repudiability is called transactional non-repudiability, in which the payee can also prove that the information was received by customer.

1.2.4 Stealability

A major issue with traditional payment systems is that the transfer of funds can be intercepted and redirected, in other words they can be stolen. Electronic payment systems must ensure that this is not possible.

1.2.5 On/Off Line

A protocol is off-line if it does not require the interaction of any third party to complete the transaction. A protocol is on-line if some third party (usually bank

or broker) must be contacted to verify the protocol. For example traditional cash protocols are off-line but credit card transactions are on-line.

1.2.6 Double Spending

Since Electronic money is just a group of bits, making copies is very easy. Since the copy is indistinguishable from the original, one might think that counterfeiting would be impossible to detect. Thus double spending is a payment instruction used twice or more, either by customer, a vendor or any third party involved in the transaction.

In case of on-line payment systems merchants prevent double spending by contacting third party for every transaction. The merchants accepts the payment only when the third party authorizes that transaction. But in case of off-line protocols double spending can be prevented by using cryptographic protocols to reveal the identity of the double spender.

1.2.7 Overspending

The customer should spend only the amount for which he was given access by the bank or broker.

1.2.8 Transaction Cost

Per Transaction cost of a payment protocol is an important aspect that determines the applicability of a payment system in a particular scenario. For vendors selling goods whose cost is very less (e.g news paper articles) credit cards are not a suitable payment protocol, because the cost of processing the transaction is more than transaction itself.

1.3 Micropayments Characteristics

Micropayments significantly differ from Macropayment systems. These differences arise from the requirements of these systems.

1. Not All micropayment transactions need be fraud free. A few illegal transactions are acceptable as long as the number of such transactions remains substantially small compared to the total number of transactions in the system.

2. Information about individual micropayments need not be maintained. This is again motivated by the fact that the micropayment is worth only a few cents.

3. Strong public cryptographic techniques must be avoided because the cost of individual transactions is not very high. The goal of security for micropayments is risk management not total prevention.

4. *Lack of Customer - Vendor Accounting* : In Micropayments a vendor deals with many casual customers. (i.e those customers who purchases once and never return again or those who make purchases with that vendor very rarely). Therefore,the vendor does not need to maintain separate accounts for customers.

5. *Efficiency* : The underlying cost of smallest transaction is a small fraction of value being transacted.

6. *Flexibility* : Scalable support for commerce applications on both intranets and the Internet with the ability to make secure payment for a variety of products and services.

7. *Speed* : The latency of transaction must accommodate the casual nature of micro consumer and the computation overhead for transaction must be low enough that the currency server can handle many transactions per second.

1.4 Objective

Our objective is to design a micropayment protocol for transfer of payments from customer to vendor through the services of a common broker which will satisfy many of the required characteristics.

We will give an outline of existing protocols in next chapter. so far there exists no fully anonymous off-line micropayment protocol. This protocol is an attempt in that direction.

1.5 Organization of the thesis

The rest of the thesis is organized as follows

1. In chapter 2, we discuss about some of the existing protocols for micropayments and we then give the characteristics that various protocols satisfying , advantages and disadvantages.
2. In chapter 3, we present the details of our approach for micropayments.
3. In chapter 4, we will give the resource and performance analysis of our protocol.
4. In chapter 5, we present the conclusions and scope for further work.

Chapter 2

Existing Micro Payment Systems

Currently there are a number of different micropayment systems. These range from proposals in research literature to systems in commercial trials. The payment systems are either on-line or off-line. NetBill[1] is an online payment protocol, where each payment order is authorized by the NetBill server. Examples of off-line payment protocols are PayWord and MicroMint[3], NetPay[4]. In PayWord there is no protection from double spending but it can be detected during the payment clearing process by broker. The commercial system Millicent[2] uses vendor specific currency where third party (usually broker) must be online whenever customer wishes to contact a new vendor.

2.1 NetBill

The NetBill payment system has three parties : the customer, the vendor and the netbill server. The transaction has three stages price negotiation,goods delivery and payment.

In a NetBill transaction, the customer and vendor interact with each other in first two phases about items to be purchased and price to be charged; the netbill server is not involved until the payment phase, when the vendor submits a transaction request. The customer communicates with the NetBill server only in the case of communication failures and for administrative functions. Customer can optionally

protect his identity also . Both customers and vendors are provided with proof of transaction results from NetBill server. Figure 2.1 shows the relationships among parties in a NetBill transaction.

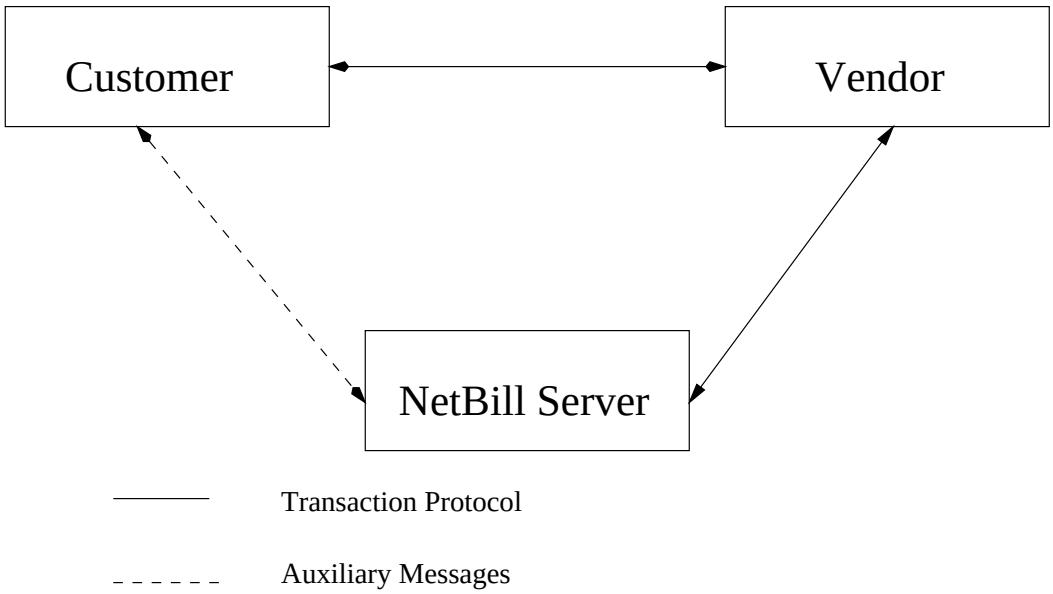


Figure 2.1: Parties in a NetBill Transaction

2.1.1 Price Request Phase

- C $\Rightarrow$ M T_{CM}, E_{CM}(Credentials,PRD,Bid,RequestFlages,TID)
- M $\Rightarrow$ C E_{CM}(ProductID,Price,RequestFlages,TID)

The identity can be pseudonym, here the customer requests a price quote of the vendor along with some optional credentials establishing her membership in groups which may be give special discounts. Bid is optional for customer to indicate his pay. The vendor stores PRD(Product Request Data) which may be later used for goods delivery phase and generates new RequestFlags which will satisfy the customer queries and indicates his price quote.

2.1.2 Goods Delivery Phase

- C $\Rightarrow$ M T_{CM}(Identity), E_{CM}(TID)
- M $\Rightarrow$ C E_K(Goods), E_{CM}(CC(E_K(Goods)),EPOID)

Now the customer requests the vendor to deliver the goods by supplying the TID, that was used in price negotiation phase. Then the vendor generates a symmetric key K, encrypts the goods with K and sends the encrypted goods, along with the cryptographic checksum computed on the encrypted goods,so that customer detects any discrepancy before sending payment order. The EPOID is globally unique identifier used by NetBill server to identify the transaction.

2.1.3 The Payment Phase

- C $\Rightarrow$ M T_{CM}(Identity), E_{CM}([EPO]$_C$)
- M $\Rightarrow$ N T_{MN}(M), E_{MN}([[EPO]$_C$,M_{Acct},MMemo,K]$_M$)
- N $\Rightarrow$ M E_{MN}([Receipt]$_{N-DSA}$,E_{MN}(EPOID,CAcct,Bal,Flags))
- M $\Rightarrow$ C E_{CM}([Receipt]$_{N-DSA}$,E_{CN}(EPOID,CAcct,Bal,Flags))

Here the customer sends the payment order which the vendor forwards to NetBill server for verification. The NetBill server after verification acknowledges the vendors, after which the vendor passes the symmetric key K to the customer. the payment order contains all the details about transaction, will be used to resolve any disputes.

2.1.4 Disadvantages

It increases the cost of transactions since the bank has to authorize every transaction. The Transaction cost is very high because of the following reasons.

- The NetBill server should be online to authorize each transaction. Hence there is high risk of system failure, traffic congestion and payment latency.

- Extensive use of Cryptography
- The NetBill server should store the payment orders.

2.2 Millicent

The Millicent system [4] introduced by Digital uses brokers as intermediaries who simplify the complexity on behalf of users and vendors. They handle account creation, maintenance and billing activities for customers and provide value added services to vendors. Brokers buy and sell vendor scrips as a service to both vendors and customers.

The Millicent system uses electronic ticket called scrip, which represents an account the customer has established with a vendor. The balance of the account is kept as the value of scrip. When customer makes a purchase, the value of the transaction is deducted from scrip's value and new scrip is returned as a change. The Millicent system assures users, vendors and brokers that scrip has not been tampered or previously spent.

The characteristics of the system are

1. The scrip is vendor specific currency whose structure is defined latter.
2. Validation of the scrip is performed in a decentralized manner which requires no support from broker
3. Strong cryptographic techniques are not used as the value of each transaction is very small.
4. Vendor does not maintain account information. It is maintained by the customer and is encoded in the scrip itself.
5. The trust model is asymmetric, where brokers considered to be most trust worthy followed by vendors and finally customers. The customers and vendors can independently check the scrip hence any fraud by broker can be easily detected. Vendors fraud is not providing goods for valid scrips in which case the brokers drop the vendors who cause too many complaints.

2.2.1 Scrip structure

The scrip has the following fields. The scrip structure is shown in fig 2.2

- Vendor identifies the vendor for the scrip
- Value gives the value of scrip
- ID# is the identifier of the scrip
- CustID# is used to produce customer secret.
- Properties are extra information about customer(age, state etc) to vendor.

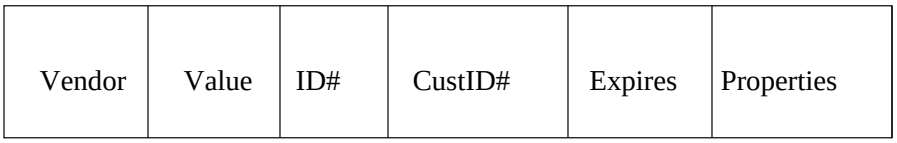


Figure 2.2: Scrip structure

2.2.2 Transaction protocol

Each customer is given a customer_secret whenever he is buying scrip. The customer sends a request for items,scrip, request signature.

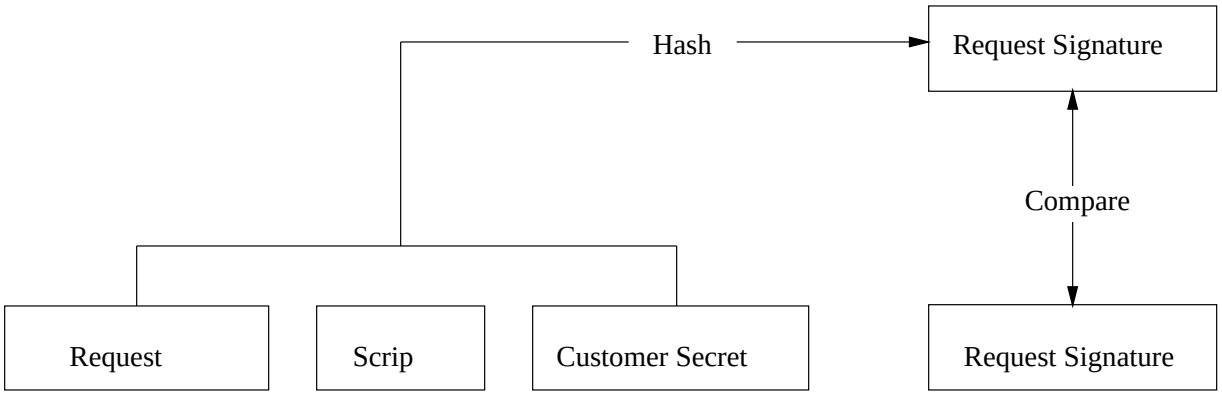


Figure 2.3: Authentication of scrip

Whenever customer spends a scrip, the value of of transaction is deducted and new scrip (with new value and new ID#) will be given to the customer. Any eaves dropper who listens to scrip cannot spend the scrip as he has no knowledge of the customer_secret. Whenever vendor receives a request he calculates customer_secret from CustID# and hashes Request, Scrip, Customer Secret to generate request signature and compares it with the request signature that customer has sent. The vendor records unique identifier of the scrip ID# of every piece of scrip that is spent, so that the customer cannot spent the scrip again. To save the vendor from from maintaining ID# forever, the scrip was given an expiry time.

2.2.3 Drawbacks

This system uses vendor specific currency, so each customer should have different scrips for different vendors. Moreover the customer should trust vendor and there is no way of identifying the vendor cheating.

2.3 PayWord

PayWord is a credit-based scheme based on chains of "paywords". The user establishes an account with broker who issues a certificate to user. The certificate contains brokers name, users name, users public key, users delivery address, expiration date of certificate and related information. The user and broker have long term relationships. The Users certificate has the form :

$$C_U = \{B,U,A_U,PK_U,E,I_U \}_{SK_B}$$

The PayWord certificate is a statement by broker to any other vendor that broker will redeem all the paywords produced by customer before the expiration date. Whenever customer wants to transact with a new vendor V, he will calculate the payword chain by picking a random w_n and then computes

$$w_i = h(w_{i+1})$$

for i = n-1, n-2 , ... 0. The root of the root of chain is w_0 which will not be treated as a coin and used only for the authentication of subsequent paywords. The Customer establishes his commitment by sending

$$M = \{V, C_U, w_0, D, I_M\}_{SK_U}$$

Here V identifies the vendor, C_U is user's certificate, w_0 is the root of the payword chain, D is the current date and I_M is any additional information that may be desired (ex length of chain etc). M is signed by Customer, which will be used to prove that customer has given the payword chain to the vendor. The vendor then verifies the customer certificate to ensure that he is a valid customer supported by broker B. This commitment authorizes B to pay V for any of the paywords w_1, w_2, ... w_n that V redeems from B before expiration date D.

2.3.1 Payments

A payment P from Customer to Vendor consists of a payword and it's index

$$P = (w_i, i).$$

The user spends the paywords in order : w_1 first, then w_2, and so on. Here the validation of w_i consists of only one hash, because the vendor already has w_{i+1}. if Customer wants to pay l paywords after spending w_k paywords, he can send simply one message with $P = (w_{k+l}, l+k)$. Then customer can has it k times to get w_k. Here the paywords are vendor-specific and user-specific. This also supports variable sized payments by specifying value of paywords in customer's commitment.

2.3.2 Disadvantages

- Use of Certificates , public key cryptography and the customer has to sign for each new vendor and vendor has to verify two certificates for ensuring the validity of the customer and payword chain.
- Since payword is a credit-based scheme where user's account is not debited until some time after purchases, this provides more opportunity for fraud since the customer can make large number of purchases with an account containing insufficient funds.

2.4 NetPay

The micropayment protocol NetPay[4] is based on PayWord system, but this is a debit-based system and it also addresses problem of overspending by customer. Broker is trusted and is responsible for all maintenance of system. The difference between NetPay and Payword is that in NetPay, the Broker generates the payword chains but in PayWord Customer generates the payword chains. Broker need not be on-line for paywords validation of first vendor but whenever customer contacts a new vendor previous vendor must be on-line. This protocol also minimizes the number of public key operations. The flow of messages in the transaction are shown in fig 2.4

2.4.1 Obtaining Payword chain

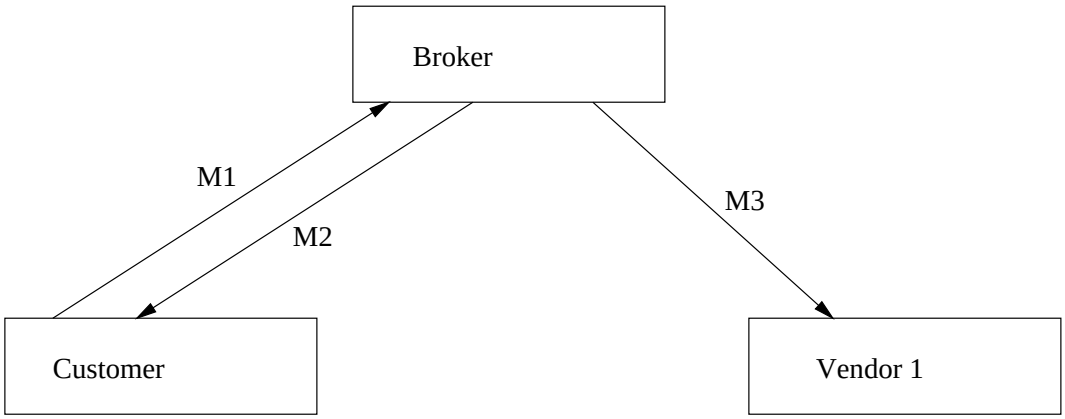


Figure 2.4: Flow of messages for receiving payword chain

Here the customer sends message M1 which consists of customer id, number of paywords, vendor address.

M1 = { ID_c,n,V1 address }

Then the broker debits money from customer account and creates the payword chain w_0,w_1, ... w_{n+1} which satisfy $w_i = h(w_{i+1})$. The broker stores w_{n+1} used to prevent the customer from overspending and forging paywords in that chain. Then the customer receives w_1,w_2, ... w_n encrypted by customer's public key and touch

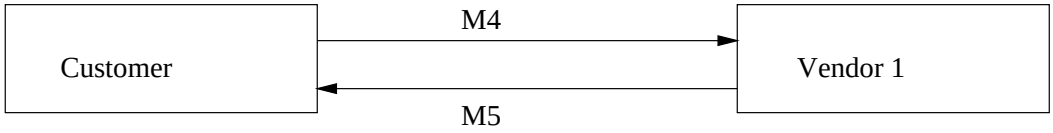
stone w_0 is passed to the vendor

M2 = { w_1,w_2, ... w_n }$_{PK-customer}$

M3 = T= { ID_c, w_0 }$_{SK-broker}$

2.4.2 Payments to First Vendor

When customer makes a purchase from V1 he sends M4 to V1

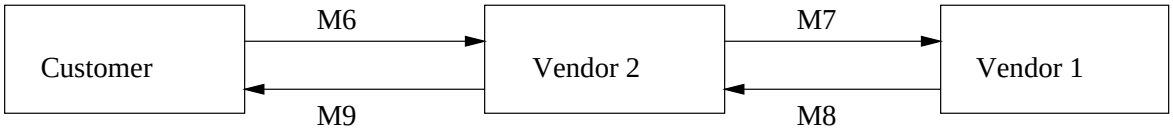


M4 = { ID_c ,P, O }

where payment P = { (w_j,j), (w_{j+1},j+1), ... (w_{j+m-1},j+m-1) }, m is the number of paywords and O is the order. Vendor 1 verifies the payword chain to see that hashing w_j for j times matches w_0. Then customer is supplied with goods and acknowledgement(M5)

M5 = { ID_{v1},receipt of payment }

2.4.3 Payments to Other Vendors



When customer wishes to purchase from a different vendor V2, then he sends message M6.

M6 = { ID_{v1}, ID_c,P,O }

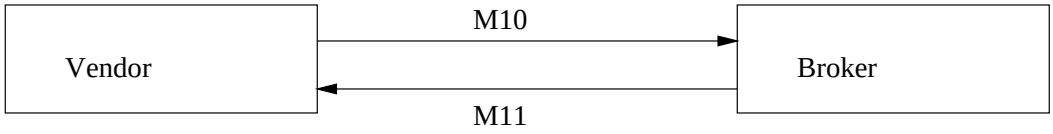
V2 transmits M6 to V1 M7 = { ID_c,ID_{v2} }

Now V1 sends the index of last payword that Customer spent with him.

M8 = { T, { ID_{v1},ID_{v2},index }$_{SK-V1}$ }

M9 = { ID_{v2},receipt of payment }

2.4.4 Vendor Redeeming from Broker



Whenever vendor wishes to redeem money from broker then he sends message M10.Then broker verifies each payword from vendor and send an acknowledgement to vendor

M10 = { ID_c, ID_v, P }

M11 = { vendor's balance amount }

2.4.5 Disadvantages

Whenever a customer wants to contact a new vendor, last vendor with whom the customer has transacted must be online, thus increasing the burden on last vendor. Thus it is not fully off-line protocol.

2.5 Summary

In this chapter we have discussed some of the existing micropayment systems. Other payment protocols include Clickshare, Internet Dollar, Jalda, NewgenPay etc. The various properties each protocol satisfy are shown in the following table.

	NetBill	Millicent	PayWord	NetPay
Anonymous	Partial	No	No	limited
On/Off line	Online	Offline	Offline	partial online
Public key cryptography	True	False	True	True
Vendor specific currency	False	True	True	Fasle
Overspending	False	True	False	Fasle

Chapter 3

Our Protocol

The RSA public key algorithm[5] can be used to make payments over the web but the signature generation and verification takes a lot of time and it is costly. Our main goal is to avoid the public-key operations required for payment by using hash operations. Hash functions are about 100 times faster than RSA signature verification and about 10000 times faster than signature generation. To support micropayments exceptional efficiency is required otherwise cost of transaction will exceed the value of items itself.

In our scheme entities are brokers, customers and vendors. Brokers authorize customer to make micropayments to vendors and vendors redeem the payments from them. This is a credit-based system where customers are charged by broker before making payments. It also address the problem of double spending and other required criteria. This is a completely off-line protocol and customers buy coins at one broker and can spend them with any vendor.

3.1 Prerequisites

We are using collision resistant hash functions and authentication trees by customers and brokers.

3.1.1 Collision-Resistant Hash Functions

A *collision resistant hash function* is a hash function [6] is a function H which has the properties of

- *compression* - H maps an input x of an arbitrary finite bit length to an out-put H(x) of fixed-length.
- *ease of computation* - given H and input x, H(x) is easy to compute.
- *preimage resistance* - for essentially all pre-specified outputs, its computationally infeasible to find any input which hashes to that output. i.e to find any preimage $x^{'}$ such that $H(x^{'}) = y$ when given any y for which a corresponding input is not known.
- *second preimage resistance* - it is computationally infeasible to find any second input which has the same output as any specified input, i.e given x, to find a 2nd preimage $x^{'} \neq x$ such that $H(x) = H(x^{'})$ and
- *collision resistance* - it is computationally infeasible to find any two distinct inputs x, $x^{'}$ which hash to the same output i.e such that $H(x) = H(x^{'}$ (Note that here there is free choice of both inputs) While the existence of collision-resistant hash functions is still an important open problem, there exists a set of fast candidate hash functions including MD5[7], Secure Hash Algorithm(SHA) etc

3.1.2 Authentication Trees

Our protocol makes use of authentication trees very similar to ones introduced by Merkle[8] for validation of various steps.
Definition 3.1 Let $h : \{0,1\}^* \rightarrow \{0,1\}^k$ be a secure hash function. An *authentication tree* is a binary tree such that

- The tree is complete, i.e., every internal node has exactly two children.

- Every node x (including leaves) has a k-bit string v[x] associated with it. Further, for any internal node x, v[x]=h(v[x_L]v[x_R]) where x_L and x_R are respectively the left and right child nodes of x.

Definition 3.2 The *authentication path* of a node string v[x^0] is defined as the sequence of strings v[x^0],v[x_s^0],v[x_s^1],v[x_s^2], ..., v[x_s^t] where x^0, x^1, x^2,..., x^t, r is the path from the node x^0 to the root r and x_s^i denotes the siblings of the node x^i in the tree.

Authentication trees can be used to validate the possession of secrets in the following way. Suppose V knows the root string v[r] of an authentication tree and C knows all the node strings. C can convince V that he possesses a particular node v[x^0] by showing to V the authentication path of v[x ^{0}]. V now has all the information to compute the root string v[r] from the given strings and thus convinces herself if C indeed possesses v[x^0]. That C cannot cheat follows from the properties of the hash function.

In our protocol we use three kinds of authentication trees : *coin authentication trees, coin merging trees* generated by customer and *customer merging trees* generated by broker.

Coin authentication tree

It is generated by each customer, as shown in figure 3.1, where R_1, R_2, ..., R_N indicate random numbers , V_8=H(R_1), V_9=H(R_2), ..., V_{15}=H(R_8) and V_1=H(V_2,V_3), V_2=H(V_4,V_5), ..., V_7=H(V_{14},V_{15}). The authentication values are based on the N/2 bit random number K sent by vendor. The authentication path contains R_{2i}, h(R_{2i+1}) if i^{th} bit of K is 0 and h(R_{2i}), R_{2i+1} otherwise.

Coin merging tree

The customer generates coin merging tree with leaves as the roots of coin authentication trees, as shown in figure 3.2. Here C1, C2,.. C8 are roots of coin authentication trees. The authentication path for coin C_i is the coin authentication path of C_i followed by values of sibling nodes of the nodes in the path from C_i to root of customer tree. The authentication for C3 is (the authentication values for C3,C4,V4,V3). Given the authentic values of C_i and

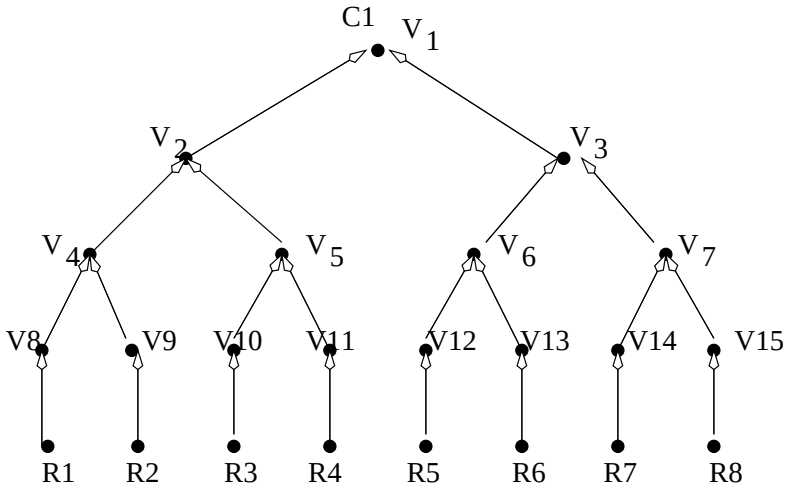


Figure 3.1: Coin authentication tree

the authentic path, any one can compute the root and verify the validity of the coin C_i.

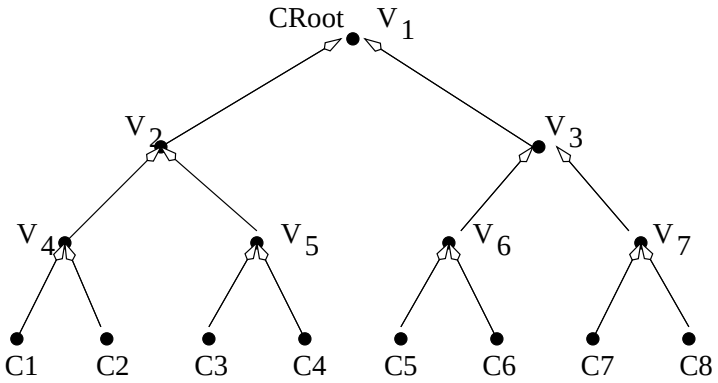


Figure 3.2: Coin merging tree

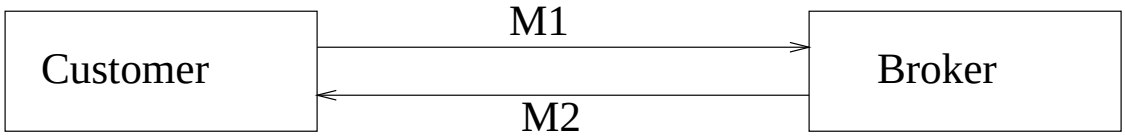
3.2 Protocol

In this section the details of the proposed micropayment protocol are discussed. Consider the trading community which consists of customer(C), vendor(V), Broker(B). The broker is honest and trusted by both customers and vendors. The customers and vendors may or may not honest. Both vendors and customers open accounts

and deposit money with broker.

The payment transaction involves customer generating coins and submit the generated coins to the broker. the broker validates them. then the customer can spend the coins with any vendor.

3.2.1 Customer-Broker



Before customer contacting any vendor, he has to first get the valid coins from the broker. The customer generates the coin merging tree and sends M1 to broker.

$$M1 = \{ ID_c, n, root\}$$

Where n is the number of coins and root is the root of coin merging tree that customer has constructed.

Then the broker will do the following two actions.

- At the end of the day, the broker debits the account of C's and constructs the customer merging tree with leaves as the roots of coin merging trees sent by customer, number of coins that customer has specified and passes to each customer the authentication path in M2. The customer merging tree is shown in fig 3.3
- He publishes the root of the customer merging tree with its expiry date. Here broker is guaranteeing to pay money to any vendor who can produce the authentication path to compute the root value.

3.2.2 Customer-Vendor

The following sequence of messages describe the transaction between customer and vendor. The customer and vendor need to agree on the amount that customer pays.

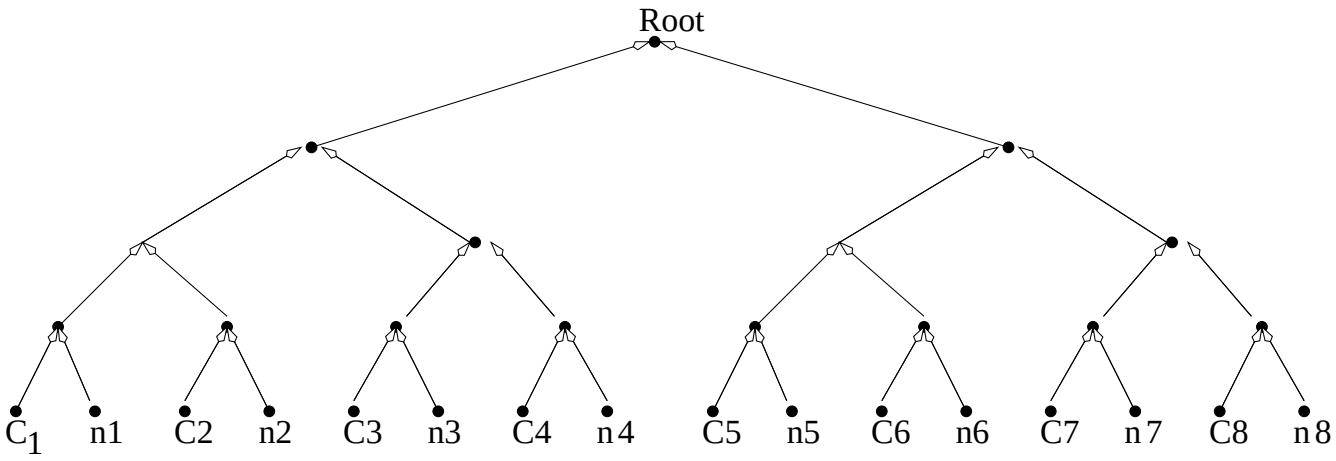


Figure 3.3: Customer merging tree

When customer makes purchase from Vendor V, the he sends M3 to V.

M3 = { ID_c, ID_b, n,(C_i,C_{i+1}, ..C_{i+n-1})}

where n is the number of coins and C_i,C_{i+1}, ..C_{i+n-1} are the roots of coins. Then vendor checks whether he supports the broker ID_b. Then he sends n random numbers in the range 0 to $2^{N/2}$ where N is the number of leaves. The coin roots ensure that customer will send the authentication paths only for these coins.

M4 = { n random numbers }

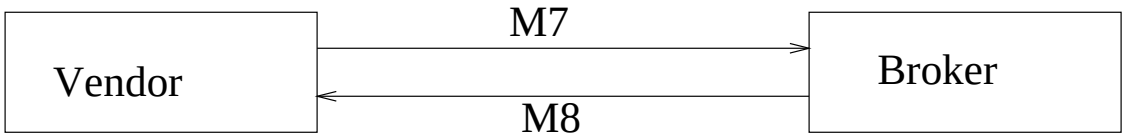
Then the customer has to send n authentication paths, one path for each random number K_j. Each authentication path consists of R_{2i} , h(R_{2i+1}) if i^{th} bit of K_j is 0 and h(R_{2i}) , R_{2i+1} otherwise and values of sibling nodes of the nodes in the path from coin root to coin merging tree.

M5 = { ID_c, n authentication paths }

Then the vendor compute the root from each authentication path to see that the coins are supported by broker. Then delivers the goods to customer in M6.

M6 = { receipt of payment, goods }

3.2.3 Vendor-Broker



For each customer, vendor sends the following message to the broker for coin redemption.

M7 = { ID_c, ID_v, authentication paths }

The broker verifies each authentication path by performing hashes on it to get the root of customer merging tree and stores the coin root. The double spending is detected, if the coin root is already stored. The double spender can be detected with a probability of 1-$\frac{1}{2^{N/2}}$ by checking the authentication paths. Same authentication paths from different vendors indicate vendor double spending and different authentication paths from different vendors indicate customer double spending. Overspending is detected if the number of coins claimed is more than what the customer has specified at the beginning. If no double spending or overspending is detected, then broker sends an acknowledgement to vendor.

M8 = { Balance of Vendors account }

3.3 Security Properties

On/Off Line

The system is completely off-line since either broker or other vendors need not be online to validate the purchases made by customer. The customer buys the coins at one broker and can spend the coins wherever he wish to, simulating real money.

Anonymous

The customer can protect his identity from vendors by using pseudo identification mechanism from broker. The broker just validates the coins from customer,but he cannot determine in advance where the customer is going to

spend the coins. It's also not possible for the vendors to identify the true identity of the customer, but the broker knows the vendors the customer has traded, when the vendors redeeming the coins that customers have spent with them. Thus broker can construct the profile of payments that customer has made. Thus it provides partial anonymity. Anonymity from broker can also be provided by using the blinding mechanism of ECash protocols. But that increases the transaction cost.

Theft of Coins

The adversary can listen to the channel and may get the coins while the customer is sending coins to the vendor. But the probability that he could spend the coins successfully is $\frac{1}{2^{N/2}}$, where N is the number of leaves in the coin authentication tree, since the adversary has to get the same random numbers that customer got in his transaction to spend the coins.

Double Spending

This is similar to theft of coins. If customer tries to double spend, he will provide different authentication paths to different vendors, which can be detected by broker. Similarly the vendor cheating can be detected if different vendors have same authentication paths.

Overspending

The broker identify that customer has overspended during the time when vendors are redeeming the coins. The number of coins(n) from each customer were used as leaves of customer merging tree. The number of coin roots greater than what the customer specified indicates overspending. Those customers found guilty can be charged heavily so as to discourage other customers from doing so.

Atomicity

The customer can send the authentication path again to the vendor, in case vendor doesn't receive the payment order. But the vendor cannot claim twice

for the same coins. Similarly the vendor can send the information again to the customer, if he receives a request again within the specified period of time.

Transaction Cost

Since this is an off-line protocol, there is no need to have central server. Moreover only the customer is generating the coin authentication trees hence the burden on brokers and vendors is very less. The processing and storage requirements analysis is done in next chapter. The storage requirements for customers can be reduced drastically by using random seed, which is also discussed in next chapter.

Non-repudiability

Non-repudiability is that the transaction cannot be denied by any of the parties involved in the transaction. The customer can compute the broker root by using the authentication path sent by broker, so broker cannot deny the customer. The vendors cannot deny the fact that customers have spent the coins with them because they have the authentication path. The customer cannot deny the payment, since only he can produce the payment order. But the vendor cannot prove that he delivered the items.

3.4 Summary

In this chapter we have introduced our scheme and described how this scheme satisfies various properties. The extension is that each broker can specify value of coins. that he also specifies the value of each root of customer merging tree , by which vendors can give specific value to each authentication path.

Chapter 4

Resource and Performance Analysis

We have implemented the proposed system to calculate the performance. The system was tested on a pentium-II MMX 233MHz processor. It is estimated that about 40000 hash functions can be computed in one second. So the processing will not create any problem. Since this is a completely off-line protocol it is more convenient for brokers and vendors.

The storage requirements for customers seems to be a bottle neck but this also can be reduced by using random seed. Since generating the random numbers is very fast, the customer need not to store the computed random numbers. He can generate them on the fly by using the random seed, whenever he wishes to spend the coins. He can store only the internal nodes in the coin authentication tree. But this will slightly increases the complexity of the system as he has to compute the random numbers for every transaction.

4.1 Processing and Storage Analysis

The time required to compute the coin merging tree and to store the entire tree including random numbers is shown in the following table. Here the number of leaves in the coin authentication tree is 8. hence the probability that double spending succeeds is $\frac{1}{16}$. The size is shown in number of KB's.

Coins	Time in sec	Size
100	0.66	60
1000	7.72	604
10000	98	6048

The verification by customer or broker is about computing 15 hashes in a 100 coin tree. it has taken about 0.02 to 0.06 sec for each coin verification. This will not increase much for trees with more leaves and even with a very large number of customers. The number of hashes required to verify will be

$$number\ of\ hashes = \mathrm{N} + \ln(M) + \ln(NC)$$

where N is number of leaves in coin authentication tree and M is number of coins that customer has, and NC is the total number of customers supported by the system.

The number of leaves in each authentication tree can be increased to reduce the probability that the double spending succeeds by increasing the number of leaves. The storage and computational requirement increase linearly but the probability will decrease exponentially. The time and storages requirements with $number\ of\ leaves = 16$ is shown in following table

Coins	Time in sec	Size
100	1.81	168
1000	20.51	1560
10000	225.43	15480

The storage requirements for the customer can be reduced drastically by storing only the coin merging tree rather than storing the coin merging tree and coin authentication trees. i.e the coin authentication trees need not be stored if we use a secret random seed. The customer stores the secret random seed and generates the necessary random values for the coin authentication paths as and when required . Thus the processing time and storage with $number\ of\ leaves = 8$ is shown in the following table

Coins	Time in sec	Size
100	0.43	7
1000	3.01	65
10000	26.13	656
100000	278.13	6456

The storage will not change even if we increase the number of leaves in the coin tree and the processing time also will not change significantly.

The processing and storage requirements with $number\ of\ leaves = 16$

Coins	Time in sec	Size
100	0.58	7
1000	4.48	65
10000	44.27	656
100000	338.13	6456

The processing and storage requirements with $number\ of\ leaves = 32$

Coins	Time in sec	Size
100	1.23	7
1000	8.18	65
10000	81.13	656
100000	643.13	6456

If Customer is storing the entire tree then

Number of nodes in each coin = N + (2N -1) = 3N - 1

Each Coin size = (3N-1)16 Bytes

Total number of nodes in coin merging tree = (3N-1)M + M-1 = 3NM - 1

Tree size = (3NM-1)16 Bytes

Hence the number of hash values to be computed to verify a coin = N+ log(M)+log(NC)

M	N	Size	Hashes
100	8	37	35
100	16	75	43
100	32	150	53
1000	16	750	46
1000	32	1500	62

If the customer is not storing the random numbers , then

Tree Size = (2M-1)16 Bytes.

But for spending the k^{th} coin one has to skip (k-1)N random numbers. Since the random number generation is very fast, one could consider this. The verification and other communication costs will not change whether the customer is storing the entire tree or not.

The communication between customer and broker is only when customer is submitting coins and if there are disputes with vendors. Similarly vendor-broker communication are during coin redemption and for any disputes.

4.2 Comparison with other protocols

The NetPay is online protocol and Millicent is using vendor specific currency. Hence we are comparing out protocol with other protocol PayWord and NetPay. The computational and storages requirements for PayWord are shown in following table

PayWord	Computational	Storage
Broker	- Sign user certificate - One hash per payment - Verify user commitment	- Store user certificate - Account of users and vendors
User	- Verify Certificates - Sign user commitments - One hash per payment	- Store secret key - Commitments & Payword chains
Vendor	- Verify all certificates - Verify all paywords	- Store all commitments - Store the accounts

PayWord uses public key cryptography, hence the transaction cost is high. The computational and storage requirements for NetBill are shown in following table.

NetPay	Computational	Storage
Broker	- Create payword chains - Sign touchstones - Verify Paywords	- Touchstones - Seeds - Accounts of V and C
User		- Payword chain - Index
Vendor	- Sign indices - Verify touchstones - Verify indices - Verify paywords	- Touchstones - Indices - Payment - Order

The computational and storages requirements for our protocol are shown in following table

	Our protocol Computational	Storage
Broker	- Create authentication tree - Verify Coins	- Authentication tree - Accounts of V and C
User	- Create Authentication tree - Calculate random numbers	- Authentication tree - Index
Vendor	- Verify coins	- Order

4.3 Summary

In this chapter we have given the results about the performance of our protocol. We also have compared the performance of this protocol with other off-line protocols

Chapter 5

Conclusions

In this thesis, we have developed a micropayment scheme for transactions over web. Micropayment schemes are important because of the many applications they have. Our protocol satisfies all the properties required for a micropayment scheme. The system can also be used for some higher *macropayment* schemes, since the probability can be reduced drastically by using a few more leaves in the coin authentication tree. It is completely off-line protocol and also it is using only hash operations.

5.1 Future Work

In this scheme we are using a common broker between customers and vendors. But the vendor can not spent the same coins at some other vendor. It would be more desirable to have a scheme where these coins can be used by others as real coins, In the sense further transactions can also be done with the same coins, still giving the chances for identifying fraud without violating any of the security properties.

Bibliography

[1] The NetBill Electronic Commerce Project, 1995 http://www.ini.cmu.edu/netbill/pubs.html

[2] Mark S. Manasse. Millicent (electronic commerce) project,1995 http://www.millicent.digital.com

[3] PayWord & Micromint, micropayment schemes,1996, R.L.Rivest and Shamir. http://theory.lcs.mit.edu/ rivest/RivestShamir-mpay.ps

[4] NetPay-An efficient protocol for micropayments on the WWW, AusWeb99, Fifth Australian World Wide Web Conference, Southern Cross University. http://ausweb.scu.edu.au/aw99/papers/dai/paper.html

[5] Ronald L. Rivest, Adi Shamir and Leonard M. Adleman(1978) A method for obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM, Vol 21(2), p120-126

[6] Alfred J.Menezes, Paul C.Van Oorschot and Scott A.Vanstone, *Handbook of Applied Cryptography*, CRC Press,1996

[7] Ronald L. Rivest(1992) The MD5 Message-Digest Algorithm,RFC 1321, Internet Activities Board

[8] Ralph C.Merkle, Protocols for Public Key Cryptosystems. In IEEE,editor, proceedings of the 1980 Symposium on Security and Privacy, April 14-16

[9] Bruce Schneier. *Applied Cryptography*, Second Edition, John Wiley & Sons,1996