Secure and Efficient Protocol for Mobile Payments

Rahul M. Godbole
National Institute of Technology Karnataka, Surathkal
PO Srinivasnagar
Mangalore - 575025
Karnataka, India
rahulmg1983@gmail.com

Alwyn R. Pais
National Institute of Technology Karnataka, Surathkal
PO Srinivasnagar
Mangalore - 575025
Karnataka, India
alwyn.pais@gmail.com

ABSTRACT

Electronic Payments have gained tremendous popularity in the modern world. Credit/debit cards and online payments are in widespread use. Bringing electronic payments to the mobile world offers huge utility for mobile users. Lack of standardized protocols, interoperability and security are major roadblocks in developing a mobile payment infrastructure. A scheme called SEMOPS (Secure Mobile Payment Service) has already been proposed by A. Vilmos and S. Karnouskos. This proposed SEMOPS architecture addresses these problems. However, it will work inefficiently for micropayments due to a lot of computation and communication for every payment. Good micropayment support is extremely important for mobile payment systems to succeed. This work focusses on enabling efficient micropayment support in SEMOPS scheme. An analysis of the security and efficiency of the proposed method is given in this paper. Our new proposed method has been found to be very efficient for micropayments in SEMOPS.

Categories and Subject Descriptors

H.4 [**Data**]: Data Encryption

General Terms

Mobile Commerce

Keywords

mobile payment, macropayment, micropayment,SEMOPS

1. INTRODUCTION

Mobile payments sytems have tremendous potential to offer many more services and much more convienience that electronic payment systems cannot offer. Mobile payment systems are already in use. However, some of them lack adequate security. They are proprietary and so are not interoperable. This lack of interoperability is the main reason why mobile payment systems have not really become popular. Some standardization which allows various competing service providers/banks to come together and offer a mobile payment service is required.

A system called SEMOPS (Secure Mobile Payment Sevice) has been proposed by A. Vilmos and S. Karnouskos[1]. This proposal has the potential to provide a secure platform where competing service providers will give secure mobile payment services. This provides a really good interoperable mobile payment system. However, payment is made immedietely in each and every transaction in this method. This method is useful for macropayments (a payment in which the payment value is so large that payment cost overhead is negligible compared to it). In case of micropayments [2] (small value payments in which the payment cost overhead become comparable to the payment value) this system will be inefficient. Efficient support for micropayments is extremely important if mobile payment systems are to become a success. This paper proposes a scheme to enable micropayments efficiently in SEMOPS architecture. A micropayment scheme has been fitted in SEMOPS architecture with no significant modifications to the original architecture. Efficiency in handling micropayments is achieved by aggregating micropayments to form a macropayment and the macropayment amount is paid thus reducing the overall cost per payment. An analysis of the security and efficiency achieved by the proposed scheme has also been done.

The rest of the paper is organized as follows. Section 2 gives an overview of what work has taken place in mobile payment systems. Section 3 describes the architecture of SEMOPS proposal and shows why micropayments will be inefficient in it. Section 4 gives the principle used by us to incorporate micropayments in SEMOPS. Section 5 elaborates on our proposal to provide efficient micropayment support in SEMOPS. Section 6 describes some security and efficiency considerations of our proposal. Section 7 gives a mathematical analysis of our proposal and we conclude in Section 8.

2. RELATED WORK

Dahlberg, Mallat, Ondru and Zmijewska [3] present a discussion on mobile payment market which gives factors to be considered from a market point of view in mobile payment systems. According to Delic N. and Vukasinovic, Ana[4] there are three different ways of designing mobile payment systems. One is using SMS based applications, the second is Point of Sale device and the third one is a mobile commerce application. The drawback in the SMS based application is

10th Int. Conf. on Electronic Commerce (ICEC) '08 Innsbruck, Austria

that the SMS will be stored on the user's phone and can be viewed by some one else. Also, it can be viewed while typing which may leak confidential information. In SMS based applications, there is a lot of trust on the mobile service provider's network. In case the SMS is in a decrypted form at some node on the network, hacking that node will reveal all user's financial information. Also, in SMS based systems, only a fixed amount of money can be transferred in order to be charged on the mobile phone carrier bill. In Point of sale applications, the mobile phone should synchronize with the merchant system. The disadvantage of this is that mobile phones will require a modification and merchants will have to install devices in their payment systems. In the third approach, the user chooses what he wants to buy and conducts transactions with a secure mobile payment system. The main advantage of this method is the remarkable convenience it gives to the user. However, present technology is not very well equipped to handle this.

Mobile communication technologies themselves pose some security challenges. Chales Brookson[5] says that GSM provides over the air security but data can be stolen at intermediate nodes in case of an intrusion there. Also, network is not authenticated in GSM. So, it is possible to set up a fraud BTS and listen to others' communication. So, we cannot rely on security provided by GSM networks.

There are solutions available for mobile payments. However, security is a big and not very well addressed concern in these systems. Karnouskos, Kauffman, Lawrence and Pousttchi [6] give research advances in mobile payment area. They show efforts that have been put in to build mobile payment systems and analyze them. Tsalgatidou, Veijalainen and Pitoura [7] present challenges in mobile commerce from a business and technology viewpoint. According to them, characteristics of wireless communication, device constraints and mobility are key challenges encountered while building mobile payment systems. Jan and Pigneur [8] assess Near Field Communication(NFC) as an option for mobile commerce. However, NFC limits the scope of usage of a mobile payment system. A NFC payment device should always be present for it. Xi and Han-ping [9] give a very high level architecture of a mobile payment system. However, this architecture is too general but does give a good picture about how mobile payment systems should look like. Fourati, Ayed and Benzekri [10] give a SET based approach for mobile payment. However, it is heavily reliant on security provided by WAP and WTLS and so not flexible enough to assimilate other technologies. Zhang, Moita, Mayes and Markantonakis [11] also propose a mobile payment system. However, it relies on biometric fingerprint identification. Mobile payment systems like Paypal and Paybox are in use. However, they are incompatible with each other. A universally acceptable payment protocol that benefits all parties at stake and having a high degree of interoperabiltiy and security is needed. Such a mobile payment system should not be bound to any technology and should make the best use of present/future technologies. SEMOPS[1] is a new system that has been proposed that caters to this need. However, since payment is done for each and every transaction, micropayments will be inefficient in SEMOPS. Good micropayment support is very essential for the success of mobile commerce. A good proposal for efficient micropayments in electronic commerce systems is in [2]. In this mechanism, there is a customer, bank and a merchant. If the fraction of number of micropayments forming a macropayment is s, and each payment is 1 cent, this mechanism says that the bank should transfer $\frac{1}{s}$ cents to the merchant for every micropayment with probability s. This results in a tremendous saving on payment cost overhead. Three different schemes MR1, MR2 and MR3 have been suggested in this paper. However, in MR1 scheme, there is a risk that some customers may be overbilled and some underbilled. In MR2 and MR3 scheme, the bank may run out of money since the rate at which it gives money to merchants is far higher than the rate at which it recieves money from customers. However, [2] gives good insight into how micropayments should be performed and this insight drawn has been used in this work.

3. SEMOPS ARCHITECTURE

SEMOPS [1] (Secure Mobile Payment Service) is a proposal to come up with a general protocol for mobile payments which is not bound to any technology and is like an open standard. SEMOPS has been designed to provide an open and interoperable secure mobile payment system. It addresses the fact that mobile devices have very limited computational power and so not many cryptographic operations can be performed on mobile phones. The working of SEMOPS scheme is given below.

3.1 SEMOPS Entities

Entities in SEMOPS architecture are as follows

3.1.1 Users

A user has to subscribe to the SEMOPS mobile payment service to be able to use it. Users involved in a payment transaction can be of two types (customers and merchants) depending on the role they play during a payment transaction. Each user will have a unique PIN which he will keep confidential with himself. User uses his PIN to authorize transactions he makes in the mobile payment service.

Customer.

This is the user who pays for a service or goods he purchases. The customer can be a mobile device (mobile phone, Personal Digital Assistant). SEMOPS is capable of handling payments on the internet too. So, the customer can also be a PC on the internet.

Merchant.

This is the user who accepts money for giving goods/services. Merchant can be a handheld device, a computer or a point-of-sale terminal.

3.1.2 Payment Processor

This is a trusted partner of the user on the mobile payment system. It is the point of contact for the user to access the system. It has confidential user information like user's bank account number and is authorized to make financial transactions on the user's account. It can also authenticate and authorize the user before giving the user access to the payment service. Communication between the payment processor and user should take place over a secure communication channel. The payment processor can transfer money from the user's account to some other account. Since the user trusts the payment processor, the payment processor guarantees that it will not misuse this right. Each payment

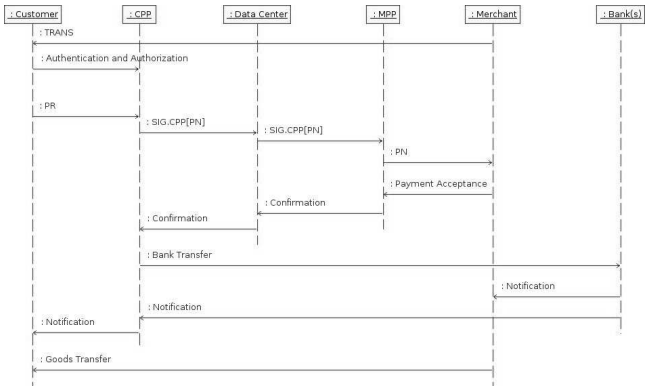

Figure 1: Working of SEMOPS Protocol

processor has its own public/private key pair. A payment processor is maintained by the user's bank. For a particular payment transaction, payment processors involved can be classified into two types.

Customer Payment Processor.

Customer Payment Processor(CPP) is the point of contact of a customer in the system. Each CPP interacts with a number of customers. Also, the customer can select a CPP which it wants to use when it wants to make a payment.

Merchant Payment Processor.

Merchant Payment Processor(MPP) is the point of contact of a merchant in the system. Each MPP interacts with a number of merchants.

3.1.3 Bank

This is a bank that has the user's account. Any number of banks can be present in the system. There are interbank procedures to transfer money between accounts in different banks.

3.1.4 Data Center

This is a central store of user and payment processor mapping. Each data center has enough data to decide which payment processor should be contacted to send a message to a certain user if the user is in that data center's coverage area. If the user is not in the data center's coverage area, the data center can decide which data center it should forward the request to, so that the message goes to the payment processor for that user. The data center is maintained together by Mobile Network Operators. Maintaining data centers also enables them to charge for their usage thus earning them revenues.

3.2 How SEMOPS Works

We will now see the working of SEMOPS payment system [1]. A list of acronyms used in the design is given in Table 1.

Figure 1 is a sequence diagram showing SEMOPS protocol. This protocol works as follows

1. The merchant (in general any POS/VirtualPOS) provides to the customer the necessary transaction details in $TRANS$. This data includes certain static and dynamic elements that identify the merchant and the individual transaction. During the whole payment process, the customer does not identify itself to the merchant, nor provides any information about itself, its bank, or any other sensitive data.

2. The customer recieves the $TRANS$ from the merchant and combines it with information that identifies himself. A standard format payment request PR is prepared. Then he selects the account manager CPP, where the payment request is to be processed. When the payment request is ready for transfer, the customer checks its content, authorizes it (using $PIN\_Customer$) and sends the PR to its CPP.

3. The CPP recieves the PR, identifies the customer and processes the PR. Processing includes verification of the availability of the necessary funds and reservation of the required amount. When the processing is completed, a payment notice PN is prepared by the CPP. It is signed by the CPP and is forwarded alongwith its digital signature $SIG.CPP[PN]$ to the Data Center of the payment service. The Data Center forwards PN and $SIG.CPP[PN]$ to the merchant's trusted payment processor MPP. In case of an international transaction however a second data center is also involved, namely the local data center of the foreign merchant's country. In general one Data Center per country is envisaged, but more than one may also exist depending on requirement.

4. The MPP recieves PN and $SIG.CPP[PN]$, verifies the signature and identifies the merchant from PN. The payment processor advises the merchant in real time about the payment by forwarding the payment notice. The merchant has the chance to control the content of the payment notice and can decide, whether to approve or reject the transaction. If the merchant confirms the transaction, a confirmation is sent by the MPP to the CPP through the data center.

Notation	Description
SEMOPS	Secure and Efficient Mobile Payment Service
CPP	Customer Payment Processor
MPP	Merchant Payment Processor
DC	Data Center
PR	Payment Request formed by the Customer
$SIG.X(Y)$	Digital Signature of X on Y
PN	Payment Notice formed by CPP by processing PR
$PIN\_CUSTOMER$	Customer's PIN
$ACCNO\_CPP$	CPP's Account Number
$ACCNO\_MPP$	MPP's Account Number
$PIN\_MERCHANT$	Merchant's PIN
$TRANS$	Transaction data. Identifies the merchant, merchant's bank and the ongoing transaction. Also contains a transaction id which will help the merchant in delivering goods to the right customer.

Table 1: List of Acronyms

5. When customer's payment processor recieves a confirmation it initiates a regular bank transfer to merchant's bank. This transfer is based on regular well-established interbank procedures. In case of successful money transfer, the merchant's bank sends a notification to the merchant, and the customer's account manager sends a notification to the customer. If for whatever reason the merchant rejects the transaction, the customer's payment processor releases the funds it has reserved for the purchase.

6. Transfer of goods can be done when the merchant get a notification from its bank stating that money transfer has been done.

A detailed analysis of this system is given in [1]. Also, as the number of customers and merchants goes up, the number of CPPs, MPPs and Data Centers can be increased in direct proportion. Thus, this approach scales well.
In this scheme, a lot of computation is done and messages are passed for each and every payment. This has an overhead cost for the payment. If the payment value (actual price if item being bought) is large, then the payment cost overhead is very small in comparison to the payment value. So, the total payment amount (payment value + payment cost overhead) is approximately same as payment value. Such a payment is called a macropayment. However, if payment value is very small then the payment cost becomes comparable to it. So the percentage addition to payment value to form total payment amount becomes unacceptably large. Such a payment is called a micropayment. So, micropayments are always made by clubbing them together such that they sum up to form a macropayment. This achieves efficiency in the payment. The exact threshold payment value above which the payment will become a macropayment has to be arbitrarily fixed to some value.

Since SEMOPS needs a lot of communication and computation per payment transaction, micropayments cannot be made in SEMOPS. This paper suggests a scheme to enable micropayments efficiently and securely in SEMOPS protocol. Aggregation of micropayments is done to make actual payments in our proposed scheme.

Before we actually see how micropayments problem can be solved in SEMOPS we will see the principle we have used to solve the problem.

4. THE PRINCIPLE USED HERE TO SOLVE MICROPAYMENTS PROBLEM

We use intermediaries between customer and merchant to pay the micropayment amount by acheiving aggregation. Let us see how this works.

4.1 One Intermediary

Consider a set of customers $C_1, C_2, C_3 ... C_m$ who pay micropayment amounts to a set of merchants $M_1, M_2, M_3 ... M_n$. We introduce an intermediary I as shown in the Figure 2. The intermediary I has an account using which it can make payments. The scheme works as follows

- Suppose C_1 wants to make a payment to M_1. Instead of paying a micropayment amount immediately to M_1 and incurring a high overhead cost, C_1 instructs I to pay the amount to M_1. This instruction to I is also a guarantee given that C_1 will redeem all the money it is supposed to pay at a later time.

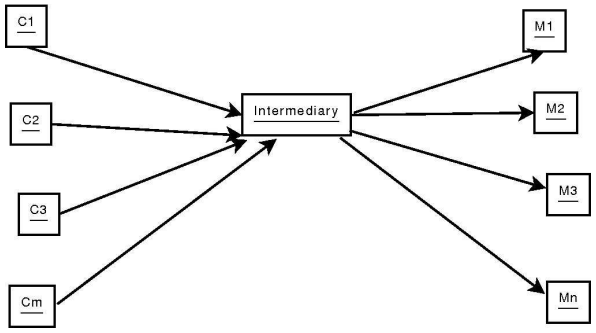

Figure 2: Aggregating Micropayments using One Intermediary

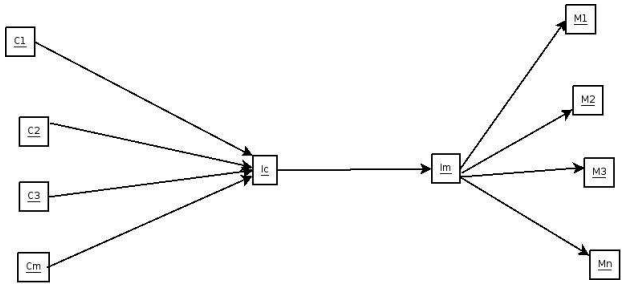

Figure 3: Aggregating Micropayments using Two Intermediaries

- If I pays the micropayment amount immediately to M_1, the payment cost overhead will again be too high. So, instead of making the payment immediately, I informs M_1 that it will pay the micropayment amount which M_1 is supposed to recieve from C_1 later. This is a guarantee given by I to M_1 that M_1 will get the money.

- M_1 then transfers goods that C_1 has asked for.

- This goes on for every micropayment transaction between any two C_i and M_j.

- I keeps a track of how much money it is supposed to recieve from each customer. Also, it keeps a track of how much money it is supposed to give to each merchant.

- After some micropayment transactions have been made in this manner, there will be a merchant M_j such that the total amount M_j is supposed to get is a macropayment. After this happens, I transfers that macropayment amount from its own account to M_j's account. Thus, since payments to merchants from I are always aggregated micropayments forming a macropayment, the payment overhead is minimized.

- After some micropayment transactions have been made in this manner, there will be a customer C_i such that

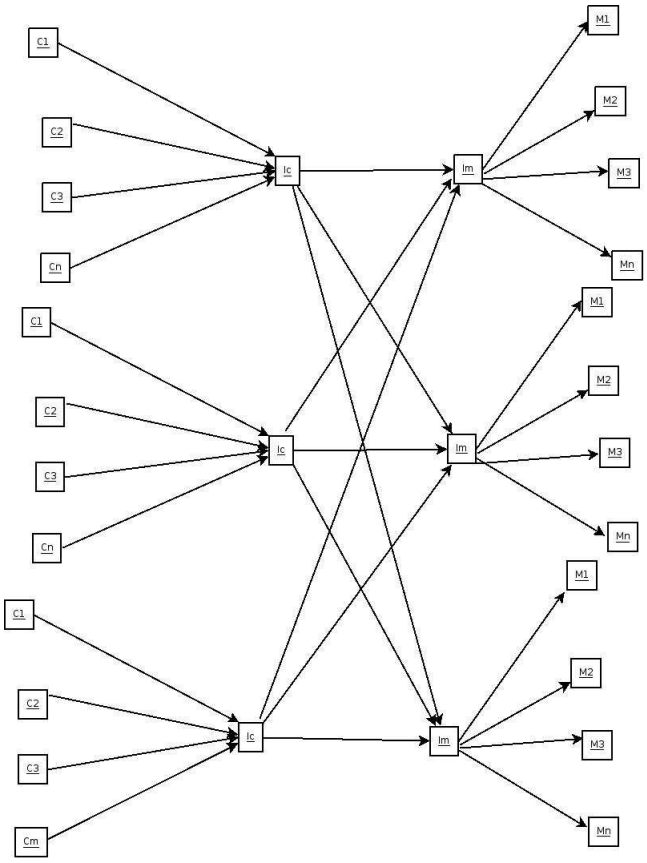

Figure 4: Many Intermediaries working with different Customers and Merchants

the total amount C_i is supposed to pay to I is a macropayment. After this happens, I gets that macropayment amount from C_i's account to its own account. Thus, since payments from customers to I are always aggregated micropayments forming a macropayment, the payment overhead is minimized.

4.2 Two Intermediaries

We can scale the above one intermediary scheme to two intermediaries.

Consider a set of customers $C_1, C_2, C_3...C_m$ who pay micropayment amounts to a set of merchants $M_1, M_2, M_3...M_n$. We introduce two intermediaries I_C and I_M as shown in the Figure 3. Customers interact with I_C and merchants interact with I_M. Intermediaries have accounts using which they can make payments. The scheme works as follows

- Suppose C_1 wants to make a payment to M_1. Instead of paying a micropayment amount immediately to M_1 and incurring a high overhead cost, C_1 instructs I_C to pay the amount to M_1. This instruction to I_C is also a guarantee given that C_1 will redeem all the money it is supposed to pay at a later time.

- If I_C pays the micropayment amount immediately to M_1, the payment cost overhead will again be too high. So, instead of making the payment immediately, I_C informs I_M to pay that amount to M_1. This is also a guarantee given by I_C to I_M that it will pay that amount at a later time.

- If I_M pays the micropayment amount immediately to M_1, the payment cost overhead will again be too high. So, instead of making the payment immediately, I_M informs M_1 that it will pay the micropayment amount which M_1 is supposed to recieve from C_1 later. This is a guarantee given by I_M to M_1 that M_1 will get the money.

- M_1 then transfers goods that C_1 has asked for.

- This goes on for every micropayment transaction between and two C_i and M_j.

- I_C keeps a track of how much money it is supposed to recieve from each customer. I_M keeps a track of how much money it is supposed to recieve from each I_C. I_M it keeps a track of how much money it is supposed to give to each merchant.

- After some micropayment transactions have been made in this manner, there will be a merchant M_j such that the total amount M_j is supposed to get is a macropayment. The moment this happens, I_M transfers that macropayment amount from its own account to M_j's account. Thus, since payments to merchants from I_M are always aggregated micropayments forming a macropayment, the payment overhead is minimized.

- After some micropayment transactions have been made in this manner, there will be a customer C_i such that the total amount C_i is supposed to pay to I_C is a macropayment. The moment this happens, I_C redeems that macropayment amount from C_i's account to its own account. Thus, since payments from customers to I_C are always aggregated micropayments forming a macropayment, the payment overhead is minimized.

- There are many customer and merchant intermediaries like these on the system as shown in Figure 4. So, after some micropayment transactions have been made in this manner, there will be an intermediary I_C such that the total amount I_C is supposed to pay to I_M is a macropayment. The moment this happens, I_C pays that macropayment amount from its own account to its I_M's account. Thus, since payments from customer intermediaries I_C to merchant intermediaries I_M are always aggregated micropayments forming a macropayment, the payment overhead is minimized.

In an actual scenario, there are going to be many customer intermediaries interacting with different customers and many merchant intermediaries interacting with different merchants as shown in Figure 4. As the number of customers and merchants increases, the number of intermediaries can be increased linearly. Thus, good scaling can be achieved.

5. THE MICROPAYMENT PROPOSAL

The proposed scheme is an addition for enabling micropayments made to the SEMOPS architecture proposed in [1].

A new micropayment scheme has been designed based on the above principle of customer and merchant intermediaries

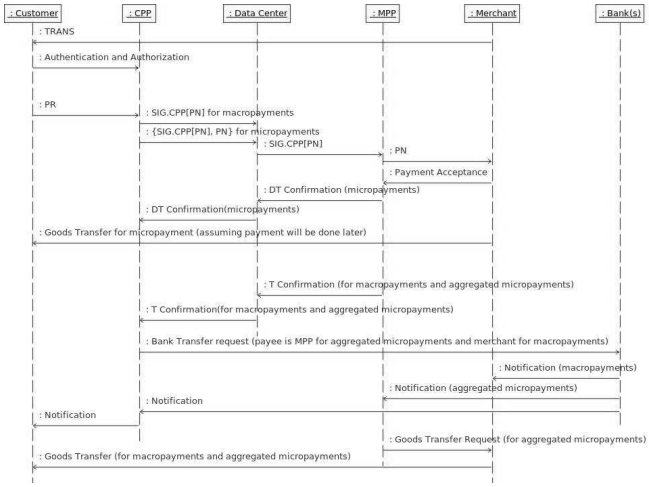

Figure 5: Micropayment Scheme

and fitted into SEMOPS protocol described earlier with minimal modifications to the original protocol. In the proposed scheme here, the CPP and MPP have an additional role to play. The CPP acts as the customer intermediary and MPP acts as merchant intermediary for micropayment transactions. The requirement is that the CPP and MPP should have their own bank accounts.

Figure 5 is sequence diagram showing the micropayment scheme. A careful observation and comparison of Figures 5 and 1 and will reveal that the mechanism for macropayments is just the same and the proposed micropayment scheme is just an addition with no changes to the original macropayment scheme. Our proposal for micropayments works as follows,

5.1 Merchant to Customer

Customer gets transaction data from merchant. Customer does not identify himself to the merchant. Merchant issues $TRANS$ to the customer.

5.2 PR from Customer to CPP

Customer builds a PR and authorizes it using $PIN\_CUSTOMER$. PR completely identifies the payment details and the customer. Customer then selects CPP. Customer authenticates and authorizes himself to CPP. It then sends PR to the selected CPP. PR also contains $TRANS$. Thus
$PR = [TRANS$, Other Required Customer Details] authorized using $PIN\_Customer$

5.3 At CPP site

CPP processes PR. It decides whether the payment is a macropayment or a macropayment. There are two possibilities.

If payment is a macropayment, it checks whether funds are available in the customer's account. If unavailable, it intimates the customer and rejects the payment. If the customer has funds, CPP reserves required amount of funds for the payment and builds a Payment Notice PN using PR. PN does not contain any customer details. It only contains TRANS and some other details that may be required.

If the payment is a micropayment, the CPP checks how much total money the customer is supposed to pay to the CPP from all previous unpaid micropayment transactions via this CPP. If this total money added to current micropayment transaction amount is less than a macropayment amount, no checking for customer funds is done. A PN is prepared as shown below. If this total money from previous transactions is equal to or greater than a macropayment amount, it means that the customer has not paid for its previous micropayment (which together form a macropayment). So, CPP initiates a bank transfer from customer's account to its account to get this pending money from previous transactions. If the bank transfer succeeds, a PN is prepared and the total amount to be paid by the customer to CPP is made zero(since all pending money has been obtained from the customer). If this transfer fails, the customer micropayment transaction is rejected. The PN built is
$PN = [TRANS, CPPDetails]$
CPP then computes a signature of this PN i.e. $SIG.CPP[PN]$.
It then forms a $PNMessage$ by clubbing together the PN and its signature $PNMessage = [SIG.CPP[PN], PN]$

This $PNMessage$ is then sent to the DC. The data center figures out to which MPP the PN should be sent to so that it will reach the merchant. It then sends this $PNMessage$ to the right MPP.

The CPP stores all micropayment requests recieved from its customers in a data structure. This data structure should be such that ,

1. It should be possible to locate all micropayment requests made by a particular customer in very less time.
2. It should be possible to know the total value of payments made for a particular customer using this data structure in very less time.

The PNs sent to different MPPs should also be stored in another data structure which satisfies two requirements

1. It should be possible to locate all PNs sent to a particular MPP very efficiently
2. The total amount from unpaid PNs for each merchant should be stored for easy retrieval

5.4 T and DT Confirmation Messages

In the original SEMOPS protocol as described above, there was a confirmation message present. Let us call this message a $TConfirmation$ message (Transfer confirmation message). When the CPP recieves this message it is supposed to transfer funds immediately. For micropayments, we add a new type of confirmation message called $DTConfirmation$ message (Delayed Transfer Confirmation Message). When the CPP recieves this message from the MPP, it should not transfer funds immediately and delay their transfer till a $TConfirmation$ message is recieved. $DTConfirmation$ message for the CPP is just an approval of the transaction and tells the CPP to pay for the transaction on behalf of the customer at a later time. How $DTConfirmation$ and $TConfirmation$ messages will be used will be clear shortly.

5.5 At MPP site

On receipt of $PNMessage$ from DC, MPP verifies the signature. If the signature is invalid, it sends a rejection to CPP via the DC. If signature is valid, MPP proceeds with

the payment.
The MPP should store these PNs it recieves in a data structure that satisfies two goals very efficiently

1. It should be possible to locate all unencashed PNs recieved from CPP i and know their total value with very less computation. This total value should be stored in $Amt\_C(i)$
2. It should be possible to locate all PNs meant for payment to a merchant and know their total value with very less computation

On receipt of PN, the MPP checks whether it is a micropayment or a macropayment by reading $TRANS$. The MPP forwards the PN to the merchant for approval. If the merchant approves the payment, following cases may occur

1. Case 1 : The PN is a macropayment:
   On receipt of approval from merchant, the MPP sends a $TConfirmation$ message to the CPP.
   $TConfirmation$ Message also contains ACCNO_Merchant so that money can be credited to the merchant's account. The CPP then initiates a bank transfer from the customer's account to the merchant's account. If transfer succeeds merchant gets a positive notification from the bank and CPP sends a notification to the customer. Goods can then be transferred.
2. Case 2 : PN is a micropayment:
   It appends the PN to the list of PNs from that CPP i.e $C(i)$. Also, it does
   $Amt\_C(i) = Amt\_C(i) + (AmountinPN)$

   There are two possibilities

   Case 2.1 : $Amt\_C(i)$ is lower than macropayment amount
   MPP sends a $DTConfirmation$ message to CPP. It also sends its $ACCNO\_MPP$ to the CPP in the $DTConfirmation$ message. This is done so that the CPP can pay a cumulative sum to the MPP when a sufficient number of micropayment transactions have been collected for merchants using that MPP. Also, it tells the merchant to transfer goods asked by the customer. It tells this to the merchant assuming that it(MPP) will be able to redeem the total value of those aggregated micropayment transactions later from the CPP when the CPP pays a cumulative sum.

   Case 2.2 : $Amt\_C(i)$ is greater than or equal to macropayment amount
   MPP sends a $TConfirmation$ message to CPP. Also, the MPP sends $ACCNO\_MPP$ in the $TConfirmation$ message. On receipt of $TConfirmation$ message from MPP, CPP initiates a bank transfer from its account to MPP's account. The MPP's bank issues a notification to MPP stating whether $AMT\_C(i)$ amount has been successfully credited to MPPs account. If the notification says that the amount has been credited, MPP tells the merchant to transfer goods asked for. If the bank notifies the MPP that the amount could not be credited, no transfer is done and the whole micropayment transaction is cancelled.

5.6 CPP recieves confirmation

The action taken by CPP depends on type of confirmation message obtained from MPP. Action taken on various confirmation messages is as follows

1. $DTConfirmation$ message: It notifies the customer that it will get goods it has asked for from the merchant and payment will be done at a later stage. Using $ACCNO\_MPP$, it keeps a track of the MPP which is supposed to recieve the payment on behalf of the said merchant.
2. $TConfirmation$ message: The following cases could occur

   Case 1: $TConfirmation$ message is for a macropayment
   In this case, the CPP requests its customer's bank to transfer money from customer's account to merchant's account.The merchant's bank issues a notification to merchant stating whether the amount has been successfully credited to merchant's account. If the notification says that the amount has been credited, the merchant transfers goods asked for. If the bank notifies the merchant that the amount could not be credited, no goods transfer is done and the whole macropayment transaction is cancelled.

   Case 2: $TConfirmation$ is for a micropayment
   This means that all micropayments intended to be made to the MPP who has sent the $DTConfirmation$ together form a macropayment. So they should now be encashed. In this case, the CPP request the bank to transfer the cumulative macropayment amount (total amount of all micropayments accumulated for that MPP) from CPP's account to MPP's account. CPP initiates a bank transfer from its account to MPP's account. The MPP's bank issues a notification to MPP stating whether the requested amount has been successfully credited to MPPs account. If the notification says that the amount has been credited, MPP tells the merchant to transfer goods asked for. If the bank notifies the MPP that the amount could not be credited, no goods transfer is done and the whole micropayment transaction is cancelled.

5.7 Dispute Settlement

All transactions taking place between all entities of the system should be logged for the record and settlement of disputes.

6. SECURITY AND EFFICIENCY CONSIDERATIONS

1. Customer trusts CPP but CPP does not trust Customer. PIN authorization by customer ensures that CPP has proof of customer transactions. This achieves non-repudiation
2. Actual payment is always done by grouping multiple micropayments. This achieves efficiency in micropayment transactions and reduces effective payment cost overhead.
3. CPP periodically transfers money it has accumulated for a MPP to the MPP. Thus MPP never has too much lag and can always have sufficient balance in its account. Hence, the MPP will never run out of money to be given to merchants.

4. CPP periodically redeems money it has spent on behalf of various customer. Thus, CPP never has too much lag and can always have sufficient balance in its account. Hence, the CPP will never run out of money to be given to MPPs.

5. Customer may not have enough funds to make a payment when it is making a micropayment. However, checking for customer funds will be an overhead. Hence, it has been avoided in the proposed scheme. So, the proposed scheme is such that the maximum amount the CPP stands to lose due to bankrupt customers not paying their amounts (aggregated micropayment sums) is the minimum value of a macropayment. In order to prevent this loss, the customer should have a certain minimum amount with its bank to use this mobile payment service. So, the CPP will always be able to get its money(since there is a minimum amount with the bank). This minimum amount should the at least the least value of a macropayment. If the customer's account balance goes below this minimum amount, the customer should not be allowed to use the mobile payment service till the time the account is replenished by the customer.

6. Suppose a customer makes a certain number of micropayments through a CPP such that his payments do not total to form a macropayment amount. After this, he never makes any micropayment using this CPP. In this case, the CPP will lose money since it does not take money from the customer's account till the customer's micropayments sum up to form a macropayment. Since the customer does not make any more micropayments, the CPP will never get this micropayment money it should get from the customer. In order to prevent this loss, CPP should take money from the customer's account after a certain cutoff interval of time even if the customer's payments do not form a macropayment. This will ensure that the CPP gets its money even if a certain customer's total payment amount never becomes a macropayment.

7. The customer can enter any amount he wants in the payment request irrespective of what was specified by the merchant. However, if this amount is incorrect, the merchant can reject it and payment will not be made. Thus, the customer cannot avoid paying the right amount of money to the merchant

8. The CPP digitally signs each PN it sends to the MPP through DC. This achieves non-repudiation . CPP (and MPP) keep records of the customer(and merchants) they interact with and are trusted by customers (and merchants). This achieves non-repudiation in the transactions between payment processors and users (customers and merchants) they interact with.

9. Each entity should start a timer when it sends a message to the next entity. If messages get lost, this timer will expire and the whole transaction should be cancelled. This will ensure atomicity.

7. PERFORMANCE EVALUATION

A mathematical analysis to evaluate the proposed scheme is presented below.

7.1 Theroretical Calculation of Percentage Cost Saving

Let m = Average micropayment amount (moving average)
M = Minimum value of a a macropayment
$x = \frac{M}{m}$ = Number of micropayments forming a macropayment at an average
C_T = Cost of money transfer from one bank account to another bank account
C_C = Cost of checking a bank account for availability of funds

Cost saving per micropayment transaction due to no checking at CPP = C_C
Average cost saving per micropayment due to aggregation between CPP and MPP = $C_T \times \frac{x-1}{x}$
So, total cost saving = $C_C + \frac{x-1}{x} \times C_T$

In the original SEMOPS scheme, there is no transfer from customer to CPP and MPP to merchant. This is a cost addition due to the proposed scheme. So, cost addition due to customer to CPP and MPP to merchant transfer is

$$\frac{C_T}{x} + \frac{C_T}{x} = \frac{2 \times C_T}{x}$$

Also, some computation and searching is done at CPP and MPP(due to data structures used to store micropayment PRs) at CPP and MPP. The cost of this is also an additional cost incurred due to this micropayment scheme. This is the cost to be paid for aggregation. Let this cost be C_d
So, total cost addition due to micropayment scheme is $\frac{2 \times C_T}{x} + C_d$

So, Overall cost saving with proposed scheme
= (Total cost saving) - (Total cost addition)
= $C_C + C_T \times \frac{x-1}{x} - \frac{2 \times C_T}{x} - C_d$

$$Overall\_Cost\_Saving = C_C + C_T \times \frac{x-3}{x} - C_d \qquad (1)$$

There are many operations in each transaction that are performed irrespective of whether the transaction is a micropayment or a macropayment (e.g. authentication and authorization between customer and CPP, signature my CPP, communication cost). Some transaction fees may also be charged by the bank over and above the fees charged due to communication and computation cost. Let the total cost of these operations be C_{min}. C_{min} is the lowerbound cost per payment (micro or macro) and no saving can be done to reduce it.

If the proposed micropayment scheme is not used, the cost of each micropayment will be
Micropayment Cost (without proposed scheme) =

$$C_{min} + C_T + C_C$$

If the proposed micropayment scheme is used, the cost of each micropayment will be
Micropayment Cost (with proposed scheme)
= Micropayment Cost (without proposed scheme) - (Cost

saving with proposed scheme)

$$= C_{min} + C_T + C_C - [C_C + C_T \times \frac{x-3}{x} - C_d]$$

$$= C_{min} + C_d + \frac{3 \times C_T}{x} \quad (2)$$

Also, percentage cost saving due to proposed scheme
= [(Micropayment cost without proposed scheme) - (Micropayment cost with proposed scheme)] / (Micropayment cost without proposed scheme) × 100
= [Micropayment cost saving sue to proposed scheme] × 100 / [Micropayment cost without proposed scheme]

$$= \frac{C_C + C_T \times \frac{x-3}{x} - C_d}{C_{min} + C_T + C_C} \times 100 \quad (3)$$

In order to get a good estimate of the performance of the proposed scheme we take sample values and analyze performance.

7.2 Analysis with sample values

Let $msgsize$ = Average size of each message that is not an acknowledgement
We neglect the size of acknowledgement messages since they are too small compared to other messages
We assume that communication cost is 2 cents/kilobyte.
So, cost of sending one message will be $msgsize \times 2$ cents.
Let s cents be the cost of computation per millisecond of computation.
RSA 1024 bit is used in SEMOPS [12]. On a 1.6Ghz Pentium celeron, we found the following computation costs for RSA 1024 bit keys Encryption : 14 millisecond
Decryption : 45 millisecond
Signature computation : 43 millisecond
Signature verification : 12 millisecond

7.2.1 Calculation of C_{min}

The non-acknowledgement messages contributing to C_{min} are

1. $TRANS$
2. Customer sends message to CPP for authentication and authorization
3. Customer sends Payment Request to CPP
4. CPP sends signed PN to the Data Center
5. Data Center sends signed PN to the MPP
6. MPP verifies payment notice's signature and sends PN to the Merchant
7. MPP sends $DT/TConfirmation$ to the Data Center
8. Data Center sends $DT/TConfirmation$ to the CPP

Thus there are 7 messages in this. Also, one signature computation and one signature verification are performed. We neglect other computation/searching since that is very small compared to signature/encryption/decryption operations.

So, C_{min} = [Communication Cost] + [Computation Cost]

$$C_{min} = 55 \times s + 14 \times msgsize \quad (4)$$

7.2.2 Calculation of C_T

The operations that take place for transferring money from A's account to B's account that we use here are as follows (assuming A and B have accounts in different banks)

1. Encryption of transfer request by A
2. A sends Transfer Request to A's Bank
3. Decryption of message by A's Bank
4. Formation of a Interbank Transfer Request by A's bank. A's bank encrypts the Interbank Transfer Request and also computes the signature of the Interbank Transfer Request.
5. A's Bank sends the encrypted Interbank Transfer Request alongwith its signature to B's Bank
6. B's Bank decrypts the Interbank Transfer Request and verifies its signature. Some more computation is performed for acceptance of funds which is negligible.
7. B's Bank sends a confirmation message to B
8. A's Bank sends a confirmation message to A

We see that 2 encryption, 2 decryptions, one signature computation and one signature verification operations performed using RSA 1024 bit keys. Also, 4 messages are passed (two transfer/interbank transfer request messages and 2 confirmation messages) .
So,

$$\begin{aligned} & C_T \\ & = (4 \times 2 \times msgsize + 2 \times 14 \times s + \\ & 2 \times 45 \times s + 43 \times s + 12 \times s) \\ & = (8 \times msgsize + 173 \times s) cents \end{aligned} \quad (5)$$

7.2.3 Calculation of C_C

1. CPP signs the checking request (to check account balance)
2. CPP sends the signed checking request to its Bank
3. Bank verifies signature on the checking request.
4. Banks gets the balance for the customer. It encrypts this balance and sends it to the CPP.
5. CPP decrypts the encrypted balance recieved.

We see here that one encryption, one decryption, one signature computation and one signature verification is performed. Two messages are passed.
So,

$$\begin{aligned} & C_C \\ & = (2 \times 2 \times msgsize + 14 \times s + 45 \times s + 43 \times s + 12 \times s) \\ & = 4 \times msgsize + 114 \times s \end{aligned} \quad (6)$$

7.2.4 Calculation of Percentage Saving

Substituting the above values of C_C, C_T and C_{min} from equations 4 ,5 and 6 in equation 3 (we neglect C_d) for percentage cost saving, we get

Percentage Cost Saving due to proposed scheme

$$= \frac{14 \times msgsize + 55 \times s + \frac{x-3}{x} \times (8 \times msgsize + 173 \times s)}{26 \times msgsize + 342 \times s} \quad (7)$$

We assume that each message is 1 kB at at average. A graph of percentage cost saving for different values of m keeping s constant is as shown in Figure 6.

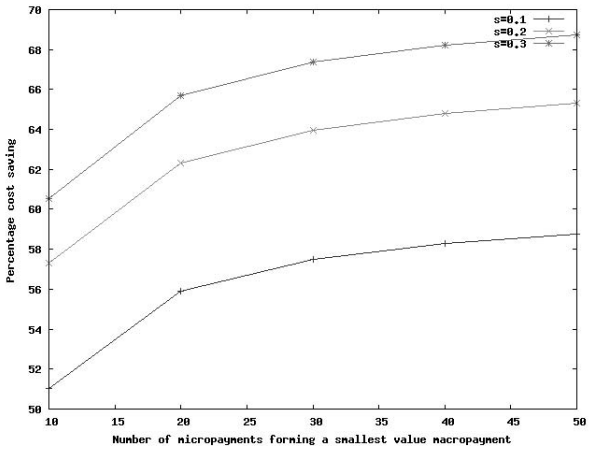

Figure 6: Theoretically derived percentage cost saving for micropayments w.r.t. x (number of micropayments forming a smallest macropayment)

We observe that as the number of micropayments forming a minimum macropayment increases, the percentage saving goes up asymptotically. That means that efficiency increases as the average micropayment value decreases but only to a certain limit.We see that the proposed scheme offers a high degree of efficiency in making micropayments.
Note that the values of communication/computation and other costs that we have taken to plot this graph are example values for evaluation. The actual efficiency achieved by this scheme in a live environment will depend on costs in that environment. However, we have shown using example (typical values) that the proposal achieves a high degree of efficiency.

8. CONCLUSIONS

We have shown that micropayments can be efficiently made using existing SEMOPS architecture with minor changes to the already proposed architecture and by adding the above explained micropayment mechanism. This architecture retains the same macropayment mechanism. Unlike many other micropayment mechanisms, the suggested approach for micropayments makes use of intermediaries to aggregate micropayments thus achieving efficiency. The suggested approach does not put the burden of any computationally complex operation on the mobile phone. This is very important in a mobile commerce environment. The micropayment protocol developed saves considerably on the amount of computation and number of messages passed per micropayment. This reduces the total cost of each micropayment.

9. REFERENCES

[1] A. Vilmos and S. Karnouskos. Semops: Design of a new payment service. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, pages 1–5. IEEE, 2003.

[2] S. Micali and R. L. Rivest. *Micropayments Revisited*. Laboratory for Computer Science, Massachussets Institute of Technology, Cambridge, MA 02139.

[3] T. Dahlberg, N. Mallat, J. Ondrus, and A. Zmijewska. *Mobile Payment Market and Research âĂŞ Past, Present and Future*.

[4] D. N. and A. Vukasinovic. Mobile payment solution symbiosis between banks application service providers and mobile network operators. In *Proceedings of the Third International Conference on Information Technology: New Generations(ITNG'06)*, pages 346–350, 2006.

[5] C. Brookson. *GSM (an PCN) Security and Encryption*. 1994.

[6] S. Karnouskos, R. J. Kauffman, E. Lawrence, and K. Pousttchi. Guest editorial: Research advances for the mobile payments arena. *Electronic Commerce Research and Applications*, pages 1–4, August 2007.

[7] Tsalgatidou, Veijalainen, and Pitoura. Challenges in mobile electronic commerce. In *Proceedings of IeC 2000. 3rd Int. Conf. on Innovation through E-Commerce. Manchester UK*, pages 1–12. Finnish National Technology Agency, November 2000.

[8] J. Ondrus and Y. Pigneur. An assessment of nfc for future mobile payment systems. In *Sixth International Conference on the Management of Mobile Business (ICMB 2007),0-7695-2803-1/07*, page 3. IEEE, 2007.

[9] H. H.-p. LI Xi. A secure mobile payment system. *Computer Technology and Application, ISSN1934-7332, USA*, 1(1):1–6, June 2007.

[10] A. Fourati, H. K. B. Ayed, and A. Benzekri. A set based approach to secure the payment in mobile commerce. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN 02, 0742-1303/02*, pages 1–2. IEEE, 2002.

[11] Q. Zhang, J. N. B. Moita, K. Mayes, and K. Markantonakis. *The Secure and Multiple Payment System Based on the Mobile Phone Platform*. Smart Card Centre Information Security Group, Royal Holloway, University of London.

[12] S. Karnouskos, A. Hondroudaki, A. Vilmos, and B. Csik. Security, trust and privacy in the secure mobile payment service. In *3rd International Conference on Mobile Business*, pages 3–5, July 2004.