Framework for Electronic Payments

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Technology

In

Information Technology

By

Dhage Manoj Madhavrao
[Roll No. 05IT6002]

Under the supervision of
Dr. C. R. Mandal

School of Information Technology
Indian Institute of Technology, Kharagpur
India
April 2007

School of Information Technology
Indian Institute of Technology
Kharagpur-721302

Certificate

This is to certify that the thesis titled "**Framework for Electronic Payments**", submitted by Dhage Manoj Madhavrao, to the School of Information Technology, in partial fulfillment for the award of the degree of **Master of Technology (Information Technology)** is a bona-fide record of work carried out by him under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of the institute and, in my opinion, has reached the standard needed for submission.

Dr. Chittaranjan Mandal
School of Information Technology &
Department of Computer Science and Engineering
Indian Institute of Technology
Kharagpur – 721302

ACKNOWLEDGMENTS

It brings me immense pleasure to express my deepest sense of gratitude to my guide **Dr. Chittaranjan Mandal** for his expert guidance and support throughout the project work. His suggestions and invaluable ideas provided the platform to the whole project work. In spite of his extremely busy schedule, I have always found him accessible for suggestions and discussions. I look at him with great respect for his profound knowledge and relentless pursuit for perfection. His ever-encouraging attitude and help has been immensely valuable.

Nothing would have been possible without the support of my family members, who have been backing me up throughout my life. I wish to convey my sincere thanks to my mother and sisters, who have given me an endless support and love and who also, provided me with the opportunity to reach this far with my studies.

I would like to express my thanks to all faculty members, my classmates and all my friends in IIT Kharagpur for their support.

Dhage Manoj Madhavrao
School Of Information Technology,
Indian Institute of Technology,
Kharagpur - 721302, INDIA.

Abstract

Implementation of an e-cash scheme on smart phone platform is described. It can work on personal computers or smart phones. This e-cash scheme is recipient specific. This particular property is useful to suppress double spending. Nevertheless it has benefits that are expected from electronic money such as anonymity. E-cash, in this scheme, is also transferable. This money scheme can be used in both, online and offline mode. In online mode, double spending is strictly avoided. In offline mode, double spending is easily detected, if attempted. In particular we have given implementation of this scheme for mobile phone devices. These devices have a lot of memory and processing power constraints. A payment scheme should work reasonably efficient on these devices. These devices provide a number of choices for wireless connectivity, such as GPRS, WLAN and BLUETOOTH. Majority of mobile phone brands also support Java. Implementation in java is discussed. Communication between mobile phone and personal computer, for carrying out payment transactions, is implemented using Java Bluetooth API’s.

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

Contents

Chapter 1

INTRODUCTION

With huge growth in electronic commerce the need for electronic money is gradually increasing. A number of notational and token based payment systems have been proposed so far to fulfill this requirement. In notational payment systems electronic communication is used to access the money stored in bank accounts. Schemes such as credit cards and debit cards [21] fall under notational payment systems. In token based payment systems digital tokens storing actual values are transferred directly between payer and payee. Schemes based on electronic wallets [3] fall under this category. The e-cash [6] scheme presented in this work is token based. Here e-cash refers to digital tokens that are transferred directly between payer and payee during payment transactions.

Key feature of electronic cash is anonymity [1] as provided by conventional cash. Notational payment systems provide banks with transactional history of payer and payee. With huge growth in electronic commerce, mobile commerce and availability of electronic devices, number of payment transactions per day has increased to a large extent. Notational payment systems are creating serious concerns regarding privacy of users through providing spending profiles and transactional history.

Electronic devices such as smart phones are available cheaply. Most of them are also able to run third party applications. Use of Internet on mobile phone device is also increasing. So a suitable payment scheme which would work on mobile phones with providing privacy is immensely needed.

1.1 Conventional payment systems

A number of payment systems are used to facilitate all kinds of business transactions. They are cash, cheques, automatic clearing house, giros, wire transfer, and debit/credit cards.

1.1.1 Cash payments

It is the most simple and popular payment system used when the buyer and the merchant are face to face. The basic advantage of cash payment is that payment is guaranteed and there is no paper work or processing involved. It does not require special equipments and all the parties involved in the transaction remain anonymous. But the problems involved in cash payments are theft and forgery.

1.1.2 Cheque payments

For transactions involving large monetary values cheque payments are convenient in terms of use. But it is a time consuming process. It involves paperwork and processing.

1.1.3 Automatic clearing house

Paper based clearing is also a time consuming process. Large number of cheques and giro payments make paper based clearing very difficult. To overcome this problem automatic clearing house (ACH) [21] has been developed. It operates similarly to paper based clearing however the payment instructions are in electronic form.

1.1.4 Giro payments

A giro [21] is an instruction to the bank of payer to transfer money to the bank of payee. The payment is not initiated unless there is enough balance in the account of the payer. So there is no overhead involved in returned giro payments. There are no written documents required, so this process is easy to conduct electronically.

1.1.5 Wire transfer

It is used for transactions involving very high monetary value. Payment procedure is slightly different due to high risk. The users of this system are corporations, banks and governments.

1.1.6 Credit card and debit card payment systems

Payment card types are credit and debit cards. Credit cards have payments from a special purpose account. This account usually has spending limit. The user need not deposit money in that account for making payments. The payments made by user are accumulated and linked to that account. Instead user is allowed to return money on installment basis. The user needs to pay interest on outstanding balance.

Debit cards are linked to the saving or checking account of the user. Whenever a user makes a payment using these cards, corresponding amount is debited from user's account and credited to the payee's account instantly. Payment can be deferred until the funds are available in the payer's account, unlike credit cards.

Fraud detection in credit card payment systems involves construction of the spending profiles. This makes it necessary to record transactional history of the user. Credit/debit card payments involve significant cost per transactions. It is generally charged to the merchant.

Credit card payments are more complex than cash payments. Issuer bank may be contacted by the merchant’s bank to authorize the transaction. Authorization involves checking validity of the card, spending limit or a maximum number of transactions allowed in a particular period. After a transaction has been successfully completed payment is cleared between issuer (bank of payer) and acquirer (bank of merchant or payee) by using infrastructure established by the card association. The acquiring bank charges the merchant for providing the payment clearing service. Electronic payment systems are generally cheaper than their paper based counter-parts.

1.2 Requirements of a payment system

Following are the typical requirements of a payment system to be widely accepted by the users and merchants, studied in [17][18].

1.2.1 Ease of use

Currently there are two payment systems widely accepted by users and merchants. They are cash and credit card payments because they are fairly easy to use. So any new payment system to be accepted by users must be as easy as these systems.

Usage of a system can be divided into three categories initial setup, transactions and monitoring or administering.

a. Initial setup may involve application download and installing or upgrading, filling out forms for respective banks.

b. Transactions involve capturing payments, refunding.
c. Administering involves accounting and verification.

A payment system should be very easy in its usage. Initial setup overhead must be least. Quantity of clicks or user typing must be minimal.

1.2.2 Expenses

Expenses to use the system should be least. Expenses involve following factors.

For Merchants

a. Initial set-up fee may be needed to buy the software and devices.
b. Monthly fee may be charged by the financial institutions.
c. Each transaction may be charged by bank due to involved processing overhead.
d. Certification fee may be charged by the Certification Authorities.

For End-users

a. In payment systems, generally users are not charged directly for making payments.
b. But depending on the technology used user may be charged.
c. If SMS or GPRS is used for communication then the service provider may charge the user accordingly.

For system providers

a. There may be deployment cost involved. E.g. deploying application over the air.
b. There is also system maintenance cost involved.

Expenses involved in above mentioned factors should be least for a payment system to be widely accepted.

1.2.3 Security

Security is a fundamental requirement of any payment system. If security is not acceptable then other cost and overhead factors make no difference. Sensitive data from all the parties must be secured during transmissions and receptions throughout the system. Some of the security related issues like confidentiality, integrity, authentication etc must be addressed.

1.2.4 Technical convenience

It includes the following.

a. System should work with all kinds of devices and heterogeneous platforms.
b. It should be network independent.
c. Performance should be acceptable always.

1.2.5 Other factors

This includes the following.

a. Reliability
b. Regulatory framework
c. Regulation
d. End-user protection.

1.3 Electronic cash concept

E-cash [4] refers to money which is exchanged electronically. Paper cash exhibits the property of anonymity so electronic payment system which is anonymous is called as electronic cash or e-cash. Anonymity and privacy are key rationales for e-cash schemes. With the help of public key cryptography [16] and digital signature (blinded as well as unblinded) it is possible to build practical e-cash systems. A user can withdraw electronic cash from bank. Bank signs the digital data representing money without getting to know the data it is signing. A user can pay to a merchant by submitting this signed data to him. Merchant verifies the signature of the bank using bank’s public key. Identified e-cash schemes can also be built. These schemes reveal the identity of the payer. E-cash schemes operate under two modes online and offline. In online mode, communication with the bank is needed to carry out payment transactions between two parties. While in offline mode, only the two parties communicate to carry out payment transactions. Bank may be contacted later on in the absence of one of the parties.

As e-cash is a form of electronic data, a number of copies can be made out of one copy. But all copies should not be spendable. Practical e-cash systems should be able to prevent or detect double spending [6]. Generally, in online mode double spending can be prevented while in offline mode double spending can only be detected.

In online e-cash systems, merchants contact payer’s bank for every transactions. The bank computer maintains a database of all the spent pieces of e-money and can easily detect if a given piece e-cash is already spent. Accordingly the merchant may accept or deny the sale.

In offline mode, e-cash systems make use of its structure and cryptographic protocols to identify the double spenders, when offline e-cash is deposited and payment is cleared from the banks. This kind of detection does not require any tamper proof system. It can be easily implemented as software and can run on any computing platform. When the e-cash is deposited in the bank, the bank can check into its database and determine if a particular e-cash has already been spent. The information gathered from all the transactions can identify the double spender. There are schemes [4] in which identity of spenders is revealed only if they double spend.

Notational payment systems allow constructing spending profiles. Misuse of personal data and spending profiles for criminal conducts has endangered lives of users of the system. E-cash can be a popular payment system addressing these issues.

E-cash should be securing from the perspective of all the actors in the systems such as user or payer, merchant and financial institutions like banks. From the point of view of the bank, a scheme is secure if it is infeasible to construct valid coins by other means than withdrawing them. From the point of view of the merchant, a coin that has been spent should always be accepted by the bank. Finally, to be secure for a user, the anonymity property should hold even if the bank conspires with other users and merchants. In addition, the bank should not be able to claim that the user has withdrawn more coins than she has, or falsely accuse the user of double spending.

1.4 Properties the electronic cash

1. No repeated spending and no forged transactions should be possible.
2. Persistence – If the personal computer of a person using e-cash crashes, all the money he is having may be lost and the person may bankrupt. But if back up is used then it should not be able to spend more than once in case of non-faulty operation of the system.
3. Anonymity – The payer should be able to spend anonymously. Otherwise organizations will be able to monitor the lifestyle of consumers. The payee or bank should be able to accept the money without being able to know from where the money is coming.
4. Transferability – Money should be able to be passed easily and anonymously.
5. Amounts – Variety of amounts should be supported.
6. Divisibility and combination – It should be able to divide or combine different amounts.
7. Cost per transaction should be low.

   It should also have following security properties.
8. From the point of view of the bank, a scheme is secure if it is infeasible to construct valid coins by other means than withdrawing them from bank.
9. From the point of view of the merchant, a coin that has been spent should always be accepted by the bank. For this effective verification system should be constructed.
10. To be secure for a user, the anonymity property should hold even if the bank conspires with other users and merchants. In addition, the

bank should not be able to claim that the user has withdrawn more coins than she has, or falsely accuse the user of double spending. E-cash should be secure in this regard also.

11. Unlinkability is the property in which given a set of coins one can not determine two coins from a same account. If a user's coins can be linkable then it can be used for identification of the user with the help of other means such as correlating payment's locality, time, size, type frequency or finding a single payment in which a user reveals his identity.
12. Non-frameability is the property that no honest user can be accused of double-spending by a corrupt bank.
13. Exculpability is the property that no user can be falsely accused of withdrawing a coin.

1.5 E-cash scheme used here

Notational payment schemes allow identity of the payer to be traced. It allows banks to construct spending profiles of users and compromise their privacy. Anonymous payment schemes are needed for maintaining privacy of the users. Cost per transaction is also high for notational payment schemes. So it's not feasible to use them for very small amounts of transactions. E-cash protocol described here addresses these problems. E-cash scheme discussed here is anonymous. It avoids double spending. Moreover it is also implemented on mobile phone platform, which makes it extremely attractive to use.

1.6 Organization of the thesis

The organization of the thesis is as follows.

Chapter 2 focuses on background and research work carried out in electronic payments field.

Chapter 3 discusses e-cash protocol which is refined and implemented in this work.

Chapter 4 discusses the design and implementation of e-cash protocol software in java. Communication using Bluetooth is also discussed.

Chapter 5 concludes the thesis and suggests future work.

Chapter 2

BACKGROUND AND RELATED WORK

Contents

Chapter 2

BACKGROUND AND RELATED WORK

A general electronic payment system involves participation of 4 actors. They are payer, payee, and their respective banks. The semantics of a payment transaction and the steps involved in it vary according to the particular payment system. In this section we will discuss classification of various electronic payment systems depending on various properties.

2.1 Classification of electronic payment systems

Following is a classification of electronic payment systems with respective different aspects studied in [19].

2.1.1 Information flow

Depending on the kind of information circulated among various parties payment systems can be classified as follows.

Direct cash like systems

These systems [4], [5], [6], [9], [12], [15] are analogous to paper cash. In these systems a user can withdraw funds from a bank in the form of digital tokens. Payer hands over the tokens to payee, who in turn deposits it in his bank. Respective banks of payer and payee then communicate to clear the payment. These tokens can be stored in personal computer of the user or his mobile phone. These transactions can also be anonymous.

Direct account based systems

These systems are analogous to conventional cheques [21]. The payer creates and signs a digital payment instruction and hands it over to the payee. Payee can then get payment honored by passing the digital payment instruction to his bank. But the main difference is that the payer and payee reveal their identity.

Indirect push systems

These systems are analogous to giro systems. In these types of systems the payer generates an instruction to his bank to transfer a particular amount to the payee's account at his bank. In these systems issuer has to be online in each payment transaction. The payee is only involved in receiving payment notifications. In these systems identity of the payer and payee is revealed to the bank.

Indirect pull systems

These systems work in exact reverse way of push systems. Payee requests payer's bank to transfer corresponding amount to his account. Payer only receives payment notifications.

2.1.2 Parties involved

The relationships between the actors involved in an electronic payment system also affect the design of the system. From buyer and seller point of view e-commerce transactions can be divided as follows.

Business-to-Consumer (B2C)

In this category seller is a business organization while buyer is a consumer e.g. retailing system.

Business-to-Business (B2B)

In this category seller is a business organization while buyer is also a business organization.

Consumer-to-Consumer (C2C)

In this category buyer and seller both are consumers. E.g. person to person money exchange

Consumer-to-Business (C2B)

In this category seller is a consumer while buyer is a business e.g. old cars buying and selling organizations.

2.1.3 Online versus offline authorization

The most important characteristic of any electronic payment system is whether authorization from the bank is needed instantly i.e. bank should be contacted immediately to authorize the payment or not. In online systems, payer's bank should be contacted to authorize any payment transaction. It puts processing load on bank servers and cost per transaction is also increased. If the communication link with the bank is down then no payment can be made. Communication link poses a single point of failure. However online authorization is essential if value involved in the transaction is very high.

In offline authorization systems, only payer and payee need to communicate to carry out a payment transaction. Bank can be contacted at the end of the day. But these systems present serious problem of double spending. Different approaches can be used to avoid double spending including smart cards, electronic wallets having observers [3].

2.1.4. Hardware versus software

Electronic payment systems also make use of tamper resistant hardware [21] devices to avoid forgery or counterfeiting. Any attempts to modify or reciprocate the data in tamper resistant hardware result in destruction of the device data and potential financial losses.

In pure software based systems these kinds of devices are not used. Other security protocols have to be provided to refrain users from getting any benefit by modifying the software or data stored or exchanged. The benefits of pure software based systems are that they are low cost. They can be ported to any computing platform.

In hardware based systems better security can be provided against traitors or counterfeiters. But the hardware is costly and need to be securely distributed and installed.

2.1.5. Size of payment

Generally there is a trade off between value in transaction and amount of security offered. That is why different payment schemes are designed for different amounts of transactions. A payment scheme designed for a particular range of transaction values. It will not suite beyond this range. Payment schemes used for carrying out transactions involving small amounts are called as micro payment and large amounts are called macro payment schemes.

In macro payments the security offered is most important due to very high value of the transaction and cost per transaction is insignificant. Very high security is offered due to risks associated with very high amounts of transactions.

2.1.6. Transaction media

Transaction medium significantly affects the design of electronic payment system. Different media used for carrying out payment transactions include Internet, mobile networks and point of sell terminals. Depending on the type of medium used for transactions different constraints are placed on the electronic payment system. Internet is a fast and cheap medium. However it can not always be used anywhere, anytime and cheaply. It is cheap for personal computers which are fixed in one location. GSM network is slow and expensive medium. But it is available on a mobile phone anytime and anywhere. Payment systems designed to work with GSM communication network must use least communication due to excessive network usage cost.

2.2 Existing electronic payment systems

A number of electronic payment systems have been proposed. In this section we will study few popular electronic payment systems.

2.2.1 Point of sale payments

Credit card payments are most popular payment system used in POS payments. Merchants check signature of the payer on the credit card for verifying payer’s identity. Magnetic stripe based credit card’s physical properties are checked to verify the validity of the card.

2.2.2 Internet payments with credit/debit cards

A many payment systems exist which can be used over Internet. However credit card payments over Internet are the most popular payment system due to ease of use. Payer hands over the payment detail such as credit card number, payer’s name and address etc to the merchant. Merchant uses this information to take an online authorization of the issuer’s bank. All payment details are communicated over SSL/TLS [33]. Main problems presented by these systems are lack of anonymity and lack of support for small amounts of payments.

2.2.3 Secure Electronic Transactions

SET [21] is used to make secure credit card payments over Internet. It provides confidentiality and integrity of the transaction data. It also provides authentication of all the parties involved in a payment transaction. Symmetric key cryptography is used for providing confidentiality and digital signatures are used for providing integrity and authentication. SET specific certification authority issues certificates to all the participants.

2.2.4 Electronic cheques and account transfers

Paper based checks are costly to process. Processing involves transport of the checks all the way to the bank on which the check is drawn. Before this it can not be deduced whether payment can be made or not. Also there is a lot of expense involved in processing the returned checks or bounced checks. Analogous to paper counterpart electronic cheque [21] contains an instruction to the payer’s bank to make a payment of a particular amount to the payee. A payer orders for goods and the payee sends an electronic bill to payer. For payment, the payer sends digitally signed cheque to the payee. Using pre-existing financial infrastructure the banks of payer and payee can clear the payment to deduct correct amount from payer's account and to credit same amount to payee's account at their respective banks. Electronic cheques can be cleared online so bounced cheques can be avoided by checking availability of funds instantly.

If the payer and the payee have accounts in two different banks then account to account money transfer process is quite lengthy. If the two parties have accounts in same online centralized bank then payment transfer is very simple. The payer connects to the online bank securely and conveys that he wants to transfer some amount to payee's account. The bank then simply has to debit the payee's account and credit payer's account. No extra financial clearing services are needed.

If anyone is allowed to open accounts then user to user payments can be easily made. There are a number of online payment systems using this principal. There are different methods to deposit money in centralized accounts. User can connect to these accounts using Secure Socket Layer. The information requested in many of these systems is very casual. Depending on the method of funding the account details of bank account number etc. are requested. The most popular method is using payment card. Any credit card can be used to make payments to this account. This bank acts as merchant accepting credit card payments and clears the payments through acquirer bank.

2.2.5 Once-off credit card number

In traditional credit cards the credit card number is kept secret. If this number is compromised then attacker can use this information with name, address etc to carry out further transactions. So the obvious way to overcome this problem is to use new credit card number for each transaction. Orbiscom [21] first brought this system to the market in their product called O-card.

Before the O-card system can be used the user must download the O-card application. Whenever a purchase is to be made the O-card application is run. The user can put upper limit on value of this transaction. After this O-card securely establishes link with card issuer system and generates new card number for this transaction. These numbers are different from the real world credit cards. These numbers have a validity period of one month. Once a transaction is made the card issuer will mark this number as invalid and refuse to process any further transaction based on this number.

2.3 Survey of electronic cash

The concept of electronic cash was introduced by Chaum et al. Chaum in [4] shows a scheme which protects payer's privacy. But if payer double spends or reuses the money, bank can trace out those payments to the account of payer and construct a proof of double spending against him. This scheme is based on suitable one way functions and modular exponentiations. But in this scheme it is not possible to transfer the payments in chains.

Trolin in [12] shows a scheme based on Merkel hash tree. In this scheme, the only computation performed by payer is evaluation of pseudorandom functions and payee verifies a signature. This scheme is computationally efficient for users. So this scheme is quite attractive for mobile platforms.

Chaum in [2] shows how to obtain blind signatures. He has several patents for it. Using this technique a user can obtain signature of the bank on a data without bank learning the data. In this way a payer can obtain signature of the bank, on a serial number without bank knowing the serial number.

Double spending is a problem that occurs in offline payment transfers. In online payment transfers bank is always involved in a payment transaction. So bank can immediately detect double spending if it occurs and hence can prevent it. There are schemes which can detect double spending at the cost of anonymity of the user. But some schemes [4] identifies double spender only if they spend more than once maintaining anonymity of the honest users.

Zamfir in [5] shows a scheme for mobile phone devices which makes use of the GPRS to connect to bank in a payment transaction, while using Bluetooth [35] for connecting payer and payee if they are within required range.

The focus of this thesis is a recipient specific e-cash scheme which has properties like anonymity, transferability, double spending prevention in online payment transfer and detection in offline payment transfer.

Chapter 3

E-CASH PROTOCOL

Contents

Chapter 3

E-CASH PROTOCOL

E-cash scheme [6] described here is token based where e-cash refers to actual digital tokens that are exchanged. Following is the formal description of the protocol.

Assume that

A = payer,

B = payee,

A = A's bank,*

B = B's bank,*

xB = B's secret key shared with his bank.

aB = account number of B.

If *A* wants to pay a sum of money *v* to *B* then following steps are followed as shown in Figure 1.

a. A initiates the transaction by sending challenge to B.
b. B replies to the challenge by sending invoice and serial number.
c. A takes the signature of the bank on the serial number makes the payment to B.

3.1 Generating e-cash for payment

B creates a nonce (random number) *n*, from this he creates following data.

$$uB = H(aB||xB||n)$$

where *H* is a suitable one way hash function [24]. *aB* is not confidential. But the secret *xB* is confidential. The bank *B** knows both. When it is provided with nonce *n* it is also able to calculate this data *uB* which is used in creation of e-cash. No one else is able to calculate this data because *xB* is secret known only to *B* and his bank *B**.

Use of nonce ensures that different value of *uB* is used each time. Different version of *uB* should be used for offline payment transfers. In that case we need to calculate *uB* as follows.

$$uB = H(g^{(aB||xB\,||n)}\ (mod\ h))$$

Where g is suitably chosen base and h is suitably chosen modulus. Along with this *B* also creates a new secret *sB*. This *sB* is used with *uB* to calculate serial number of the e-cash as follows. sB is used because nobody else should be able to deposit this cash in B's account.

$$p = H(uB||sB).$$

B blinds this serial number *p* to *p'* and sends it to *A* to get signature from its bank *A** for a monetary value *v*.

The bank has a different private key for each denomination. It signs the blinded serial number with appropriate key $(d = KR_A?\ ,\ v)$ for a particular denomination. This amount is deducted from the account of *A*. *A* then sends this signed blinded serial number to *B*. *B* can now unblind this to get original serial number with signature and forward this payment to another person *C* or deposit in his bank *B**.

3.2 Blinding the payment

Anonymity is one of the important features of e-cash. When *A* gets signature on the serial number received from *B*, *A*'s bank *A** can note down the serial number. Later when *B*'s bank *B** sends this serial number to *A** it can trace out that payment to the account of *A*. If *A** and *B** collaborates then it can be traced that *A* has made a payment to *B*. To ensure anonymity, *B* can use blinding [2] of serial number. Blinding technique is proposed by Chaum. Blinding is a kind of transform. For any appropriate random number *r* and a private key KR_d there is a function blind_r and unblind_r such that

$$\text{unblind}_r\ (\{\text{blind}_r(p)\}_{KRd}) = \{p\}_{KRd}$$

This illustrates that we can obtain an indirect signature on a data *p* by first blinding *p* to *p*\` then taking signature as $\{p`\}_d$ and finally unblinding the signature to get $\{p\}_{d.}$ This property can also be used in chaining i.e. unblind_r ? unblind_s is an inverse transform for blind_s ? $\text{blind}_{r.}$

Actual implementation for RSA signatures is as follows. Suppose *m* is the modulus of the public key of the bank from which signature is to be obtained. Create a random blinding factor *r* such that *r* is relatively prime to *m*. Then create an unblinding factor *u*, which is multiplicative inverse of *r mod m.*

1. The serial number randomly created is blinded as follows.

   $Serial\#*r^{\wedge e2}(mod\ m)$. Here assume that e_2 is the exponent of the public key of bank for rupees 2 denomination.

2. The bank signs the coin with rupees 2 secret key d_2.

   $(Serial\#.\ r^{\wedge e2})^{\wedge d2}\ (mod\ m) = serial\#^{\wedge d2} * r^{\wedge e2\wedge d2}\ (mod\ m) = Serial\#^{\wedge d2} * r\ (mod\ m)$

   Here the bank cannot record *serial#* because it does not know *r*.

3. The user after receiving this value multiplies by unblinding factor *u* which is multiplicative inverse of *r mod m*.

   $Serial\#^{\wedge d2} * r *u\ \ \ (mod\ m)$

$= Serial\#^{\wedge d2} * 1\ (m*od\ m)$

$= Serial\#^{\wedge d2}\ (mod\ m)$

In this way anonymity can be achieved.

3.3 Identification of payee

Serial number is forwarded to *A* as follows. *A* initiates the protocol by sending a nonce n_2 to *B*. *B* is expected to have a public and private key pair and a certificate issued from his bank. *B* then sends this nonce n_2, invoice *inv* containing amount to be paid, blinded serial number *p'* and hash of all this information to *A*, encrypted with his private key.

$\{n_2, p', inv, H(n_2||p'||inv)\}_{KRB}$

Where K_{RB} is the private key of *B*. Here identification of the correct payee is important otherwise payer will have to face financial losses. Payer sends a challenge to payee and verifies response to that challenge signed by private key of the payee. Payee's certificate will authenticate the public key owned by the payee.

A decrypts this data with public key of *B*, $K_{UB.}$ *A* then verifies nonce and the remaining data with hash of the data for integrity. From invoice A can deduce the amount to be paid. A then sends this data to his bank to take signature of his bank *A** and sends the signed serial number back to *B*.

3.4 Encashment of payment

When *A* sends the payment in the form of signed serial number, *B* can deposit the payment by sending it to its bank *B**. We know that payment

$P = <p,\{p\}_d>$, where $d = KR_{A?\ ,v}$ and

$p = H(uB||sB), uB = H(aB||xB||n)$

Now *B* needs to send necessary information to his bank *B**, in order to establish his own credentials and prove knowledge of the currency. *B* sends following information to bank *B** along with the signed serial number.

$$<H(xB), \{uB||n||sB\}_{xB}, H(uB||n||sB||xB)>$$

B does not send his identity on network in this protocol. Then how does the bank *B** identifies *B*? Recall that *xB* is the secret shared between bank *B** and *B*. *B** also maintains a sorted list of *H(xB)* values of all the clients. So that whenever it gets any *H(xB)* value it can map it to the sorted hash table and get the identity of the corresponding client. *B* sends *uB* to bank so that bank can verify knowledge of *(aB||xB||n)*.

B also needs to send nonce *n* and secret used for particular payment *sB* in secret, to let the bank form the serial number from its own calculation of *uB* and *sB* and to check the signature on the serial number. *B* sends this information encrypted by using *xB* as the key. The third value in the tuple represents the hash of all the values to ensure the integrity of the rest of the information. *xB* should also be included to the third field of the tuple. Because *H(xB)* can be intercepted and replayed later.

At bank *uB* is checked against its own calculation of *uB*. From *uB* and *sB* it calculates the serial number *p* to verify signature on the serial number. If *B*!=A** then *B** needs to send information *uB* and *sB* to *A** to get the payment from *A**. *A** then verifies the signature on the serial number and clears the payment. Bank *A** needs to keep track of already spent coins. Whenever it gets a new coin it should check it against the database of already spent coins to avoid double spending. To avoid storing this information infinitely the coins can be assigned an expiry date. If the coin is valid then A^* clears the payment and marks the coin as spent. A valid coin should have following properties

1- It must have been signed by denominational keys of the bank
2- It must have an expiry date later than the current date.
3- It must not appear in the database of spent coins.

The generating bank, A^* verifies $p = H(uB||sB)$ and signature on p. It should be supplied with uB and sB separately rather than just p. Because blindly checking signature on p is not safe. It is vulnerable to following kind of attack. If RSA signatures are used, then choosing $p = s^e \ (mod\ h)$, where $e = KU_{A?\ ,v}$ ensures that it will have s as its RSA signature.

After successful verification corresponding amount is transferred to B^* and B^* then deposits the amount in account of B. But A is anonymous to all.

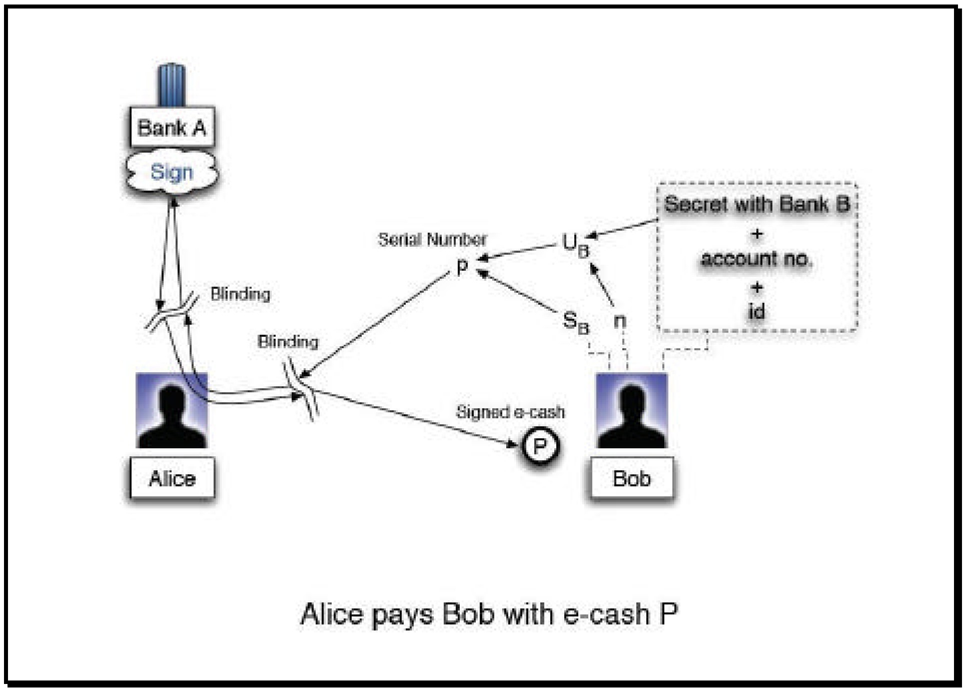


Figure 1: Alice pays to Bob

3.5 Transfer of payments

If *B* wishes to transfer this payment to *C* rather than depositing it in bank then following procedure is followed as shown in figure 2.

C generates a serial number $q = H(uC||sC)$, where *uC* and *sC* are defined analogous to *uB* and *sB* respectively. *C* blinds *q* to *q'* and passes *q'* to *B*, by following the same procedure in which *B* send serial number to *A*. *B* sends previously obtained payment *P* and *q'* to bank *B** for getting signature for appropriate value. *B** has to mark *p* as transferred or spent and *Q* as bearing value of *P*. where *Q* is signed version of *q*. If value contained in P and q is not the same the corresponding amount is adjusted from the account of B.

3.5.1 Online transfer

If *B* is online with its bank *B**, then it forwards *P* to *B**. *B** checks whether *P* has already been spent or transferred. If it is valid then it marks *P* as transferred by updating its database and verifies the proof of knowledge of *(uB||sB)* or *(aB||xB||n)* and signs *q'*, which is blinded version of *q*, for holding value of *P*. *B* then gets signed *q'* and passes it on to *C*. Finally *C* unblinds it to get *Q*. Here the bank *B** can prevent *B* from spending more than once. If *P* bears the signature of a bank other than *B** then it has to approach that bank to clear the payment *P* which it is transferring to *Q*. this ensures two important properties.

1. Payer and the payee achieve the transfer without getting to know each other's serial number.
2. No double spending can occur in online transfer.

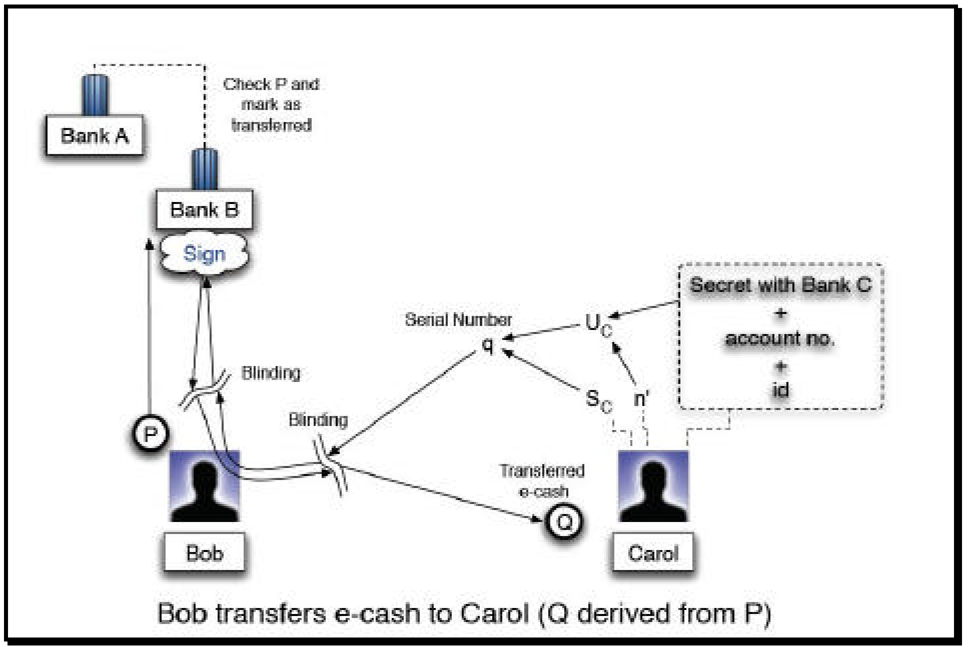

Bank A
Check P and mark as transferred
Bank B
Sign
Blinding
Blinding
P
Serial Number
q
U_C
S_C
n'
Secret with Bank C + account no. + id
Transferred e-cash
Q
Bob
Carol
Bob transfers e-cash to Carol (Q derived from P)

Figure 2: Bob transfers e-cash to Carol online

3.5.2 Offline transfers

If *B* is not online with his bank, then he can not communicate with bank *B**. So *B*'s bank *B** can not assure that he is not spending more than once. In offline payments double spending can not be prevented. This scheme detects double spending if it occurs and identifies the double spender. The scheme is illustrated in Figure 3. Here the bank of the payer will get to know the identity of the payer.

If *B* is not online with his bank then it is not enough to just send *uB* and *sB* to payee. Because it is not enough to identify *B*. Identification of *B* is needed in case he double spends. As mentioned before, different version of *uB* should be used in offline payments. It is computed as

$$uB = H(g^{(aB||xB||n)} \ (mod\ h)).$$

But this *uB* should be certified by the bank. The reason will be clear later. So *B* now passes on *P*, *UB* (certified *uB*), *sB* and a zero knowledge proof [14] to C. Later *C* can pass *P, sB, UB* and this zero knowledge proof to bank of *B* i.e. *B** to get required signature on *q*.

3.5.3 Zero knowledge proof

Consider a line $y = mx+e$, where m is a secret and e is a suitably chosen intercept. If the owner of the secret is challenged with x_0, then he can respond with $y_0 = mx_0 + e$. In this case the challenger can only verify y_0 if he knows m and e. However, if the exponents $M = g^m \ (mod\ h)$ and $E = g^e \ (mod\ h)$ are made known to the challenger, the challenger can verify that $Y_0 = g^{y0} = M^{x0}E \ (mod\ h)$, without needing to know m (or e).

For transfer of payment, we need a zero knowledge proof for m where $m = (aB||xB||n)$. Assume that $m` = aB||xB$ and $n$ be represented in $l$ bits, then $(aB||xB||n) = (aB||xB)2^{l} + n = m`k+n$, where $k = 2l$, which is constant.

We have $y_0 = (m`k+n)x_0 + e = (m`kx_0)+(nx_0+e)$.

Thus, $Y_0 = M`^{kx0}(g^{n})^{x0}E \ (mod\ h)$ where $M` = g^{m`} \ (mod\ h)$

$M`$ ,$g^{n}$ ,$E$ and $d$ are revealed to $C$ for use in checking zero-knowledge proofs. $M`$ may be required to be signed by bank. Otherwise B can reveal any spurious M' to C. This signature can be taken before the payment. As m' is a fixed quantity, M' will also be fixed quantity. So once this signature is taken, it can be used a number of times. So C can verify B's knowledge of $(aB||xB||n)$ without getting to know $(aB||xB||n)$. C now can also verify $p = H(uB||sB)$. Later when C becomes online with B^* it can pass this proof on to B^* to get the required signature.

Everything that is done in the online transfer method also needs to be done in the offline transfer. The only difference is that transfer signature from the bank is taken later, when the bank becomes online and then verification of the knowledge of $(aB||xB||n)$ is done by replaying the zero knowledge proof to the bank B^*, in the absence of the payer B. In offline payment value of P and q should be the same because payee can otherwise obtain more money than he deserves.

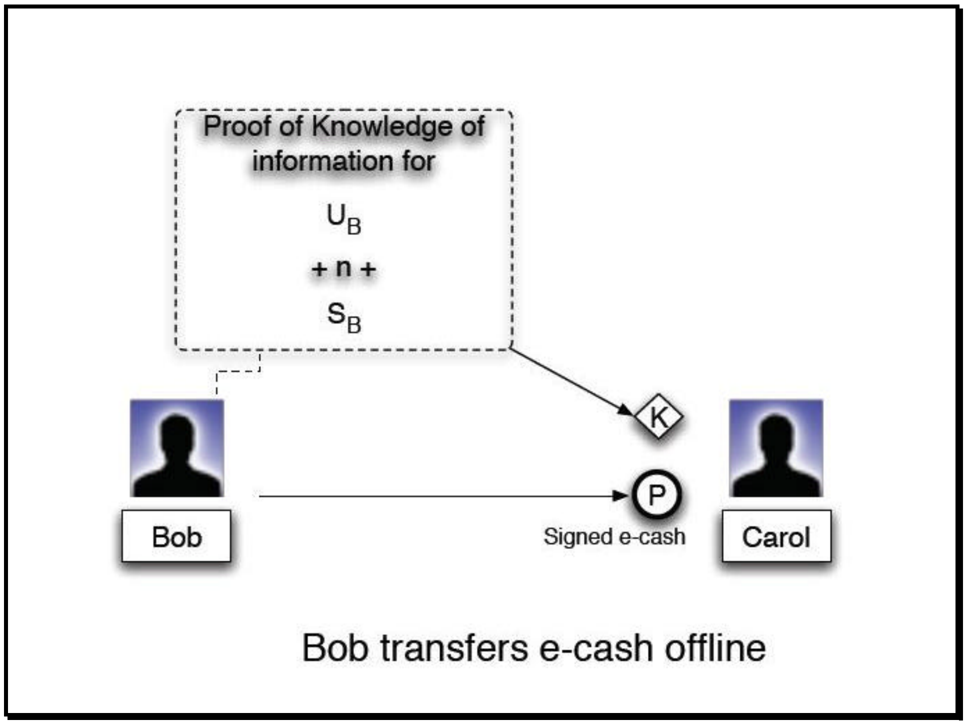

Proof of Knowledge of information for
U_B
+ n +
S_B
K
P
Signed e-cash
Bob
Carol
Bob transfers e-cash offline

Figure 3: Bob transfers e-cash offline

3.5.4 Identification of the double spender

In online transfer process *B* is identified and corresponding currency is checked for double spending instantly. So *B* can not double spend without making use of offline payments. In offline transfer, $m' = aB||xB$ is fixed and $M' = g^{m'}$ *(mod h)* is also required which is essential for offline payments. For all the account holders *(X)*, the bank can form and store tuples of their $m'(X) = (aX\ ||xX)$, $M'(X) = g^{m'}(X\ )$ *(mod h)* and *aX* values. The bank can, therefore, efficiently associate a received *M'(X)* with the corresponding *aX* and hence, the account holder *X*. Double spending occurs if the bank is requested to honor a payment for a currency with a serial number that it has already either credited or transferred. In either case the bank has the *M'(X)* (or additionally the *aX* which is account number of *X*) value of the double spender *X*, for the *uX* and also *p* values of the doubly spent currency. Thus, if a double payment does occur then this scheme will definitely identify the double spender with the help of his bank.

3.5.5 Safety of offline payment transfers

It was mentioned that the value of uB, needs to be properly signed. This is because anyone who has received a transfer payment from B knows M' and can generate new values of uB. This enables the recipient to manufacture serial numbers for currency which can be used for spurious payments that can be traced back to B. This can be prevented as follows. uB is signed by B with a special signature key $d_{T,B}$ for such transfers. This signature is again signed by $B?$. While signing the bank needs to be sure that it is signing uB for B. The bank cannot be shown n, until the time the money is encashed. Therefore, $B?$ injects the identity of B by signing $\{uB\}_{(dT,B)}M'$. M' embeds the identity of B. To prevent replays of this signature B now ties up the serial number p of the currency to be transferred to the blinded value of serial number q' of the new currency, by signing pq' as $\{pq'\}_{(dT,B).}$ C unblinds this to get $\{pq\}_{(dT,B).}$ This signature from B certifies that q was derived from p. No one else can produce such a signature and so this prevents spurious transfer currencies from being manufactured and circulated.

3.6 Features of this e-cash scheme

1. Double spending is prevented in online scenario and detected with identification of the double spender in offline scenario.

2. Money is stored in secondary storage in mobile phones and computers so there is no loss of money once it is stored.
3. This payment scheme is anonymous.
4. E-cash in this scheme is also transferable.
5. It is not feasible to construct a valid coin by any other means than withdrawing them from bank. Because it is not possible to construct private key of the bank.
6. In case of double spending merchant can deposit the coin into the bank and payment can be honored to him depending on the policy of the bank.
7. In fully online systems if the bank server is down then no payment can be made. This presents a single point of failure. But as offline payment transfers are possible in this scheme, single point of failure can be avoided.

3.7 Comparison with other e-cash techniques

Following table shows comparison of e-cash scheme described here with number of other e-cash schemes.

Properties / **E-cash Schemes**	Anonymity or Privacy	Double Spending Prevention	Transfer -ability	Forgery	H/w Security	Divisi- bility
Chaum [4]	Y	Y	N	N	N	N
Trolin [12]	Y	Y	N	N	N	N
Zamfir [5]	Y	Y	N	N	N	N
Okamoto [15]	Y	Y	N	N	N	Y
Petersen [9]	Y	Y	N	N	N	N
This E-cash [6]	Y	Y	Y	N	N	N

Table 1: Comparison of e-cash schemes

Chapter 4

IMPLEMENTATION OF E-CASH

Contents

Chapter 4

IMPLEMENTATION OF E-CASH

Protocol software is designed in java [26]. Payer or payee can be a personal computer or a mobile phone. Payer here is implemented as a mobile phone client connecting to payee which is a computer server running on merchant's behalf. Communication between payer and the payee has been implemented through Bluetooth. Following is the application architecture.

4.1 Application architecture

The application is designed to work with mobile devices and personal computers (Figure 4). Basically there are four actors in this system. They are payer, payee, payer's bank, payee's bank. Payer and payee may be using service of the same bank. Whenever a user does some shopping in a shopping mall or big store, she can make payments using her mobile phone. Merchant will have a computer server for billing and payment purpose. Payer can use her mobile phone to connect to the merchant's payment server through Bluetooth for making payments. The payer here is assumed to be a user having mobile phone. This part of the application has been developed in J2ME [27]. While the payee is assumed to be a merchant having high end computer server.

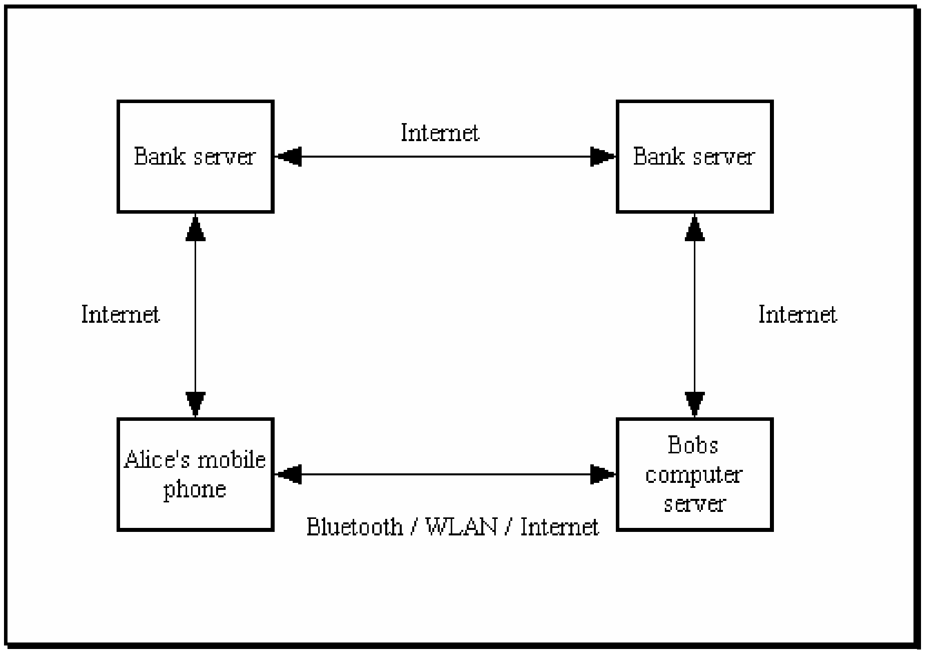


Figure 4: Application architecture

This part of the application has been designed in J2SE [26] as a concurrent server accepting connections through Bluetooth. Java is chosen because of following features.

4.2 Features of J2SE and J2ME

1. J2SE and J2ME provide secondary storage facilities for permanent storage of data on computers and mobile devices respectively.
2. J2SE and J2ME also provide interfaces to cheaper communication technologies like Bluetooth.
3. Java virtual machine makes the underlying hardware and operating system transparent to user. Same client software can be used on all MIDP supporting devices.
4. MIDlet i.e. J2ME program can be downloaded and installed on a mobile phone very easily in a single procedure. E.g. a MIDlet can

be downloaded by accessing corresponding URL associated with it like http://www.mobileapps.com/ecash.jad.

5. Though J2ME officially does not provide any cryptographic library, third party cryptographic libraries [31] are available which are widely accepted.
6. MIDlets can run in standalone mode which requires low communication with server, reducing demand for network bandwidth. Unlike browser dependent applications.
7. **7.** J2SE and J2ME provide a multithreading environment which improves resource usage and user experience.
8. **8.** J2SE and J2ME also provide API's for Bluetooth [29] and SMS [30] in addition to support for http for communication with other applications.

Payer and Payee also need to communicate with their banks. Communication between payer and payee can be achieved using following technologies.

a. Wireless LAN (IEEE 802.11)
b. Bluetooth
c. Infrared
d. Cellular networks (GSM / 2.5G / 3G)

Bank side software is implemented as a concurrent J2SE server. Payer and payee can connect to bank over Internet. But for simulation purpose we have implemented communication between payer/payee and banks entirely through Bluetooth. However, the application design is independent of any communication standard used.

4.3 Bluetooth connection

Figure 5 shows the outline of the Bluetooth communication between payers and payee.

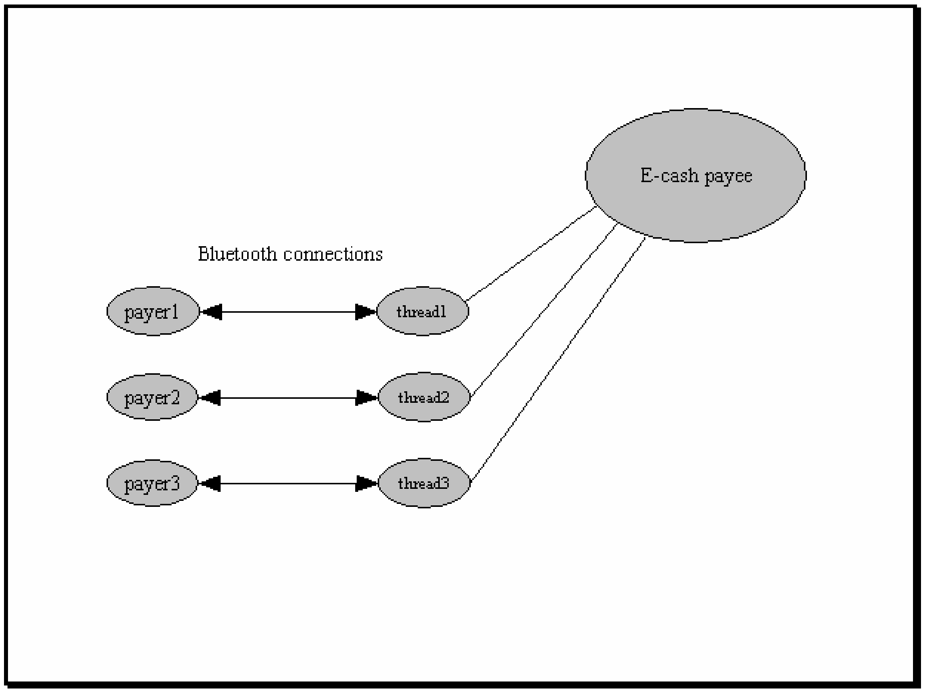


Figure 5: Bluetooth Communication

Payer and payee can transact using Bluetooth if they are in required range. Bluetooth serial port profile is used for communication between them. One of them has to be a Bluetooth server and the other has to be a Bluetooth client. Here payee is implemented as a concurrent Bluetooth server (Figure 5), while payer is implemented as a Bluetooth client. Payee can accept payments from a number of payers concurrently. Maximum number of concurrent Bluetooth connections allowed for a device can vary from 1 to 7. *B* (payee) registers and starts a service. *A* (payer) starts enquiry, finds the service and gets connected to *B*. Java Bluetooth API's (JSR 82) [29] are used for implementing Bluetooth client and server.

Bluetooth server is implemented as follows.

```
bthServer.startService();
```

startService() [20] method starts a service with provided uuid and service name and waits for connection requests from clients. Bluetooth client is implemented as follows.

```
Try {
        conn = bthClient.searchService();
        in = conn.openInputStream();
        out = conn.openOutputStream();
}catch(IOException e) {
System.out.println(e);
}
```

SearchService() method searches for a service with provided uuid and service name. If it finds the required service it opens a connection.

4.4 E-cash protocol

Figure 6 shows outline of e-cash protocol.

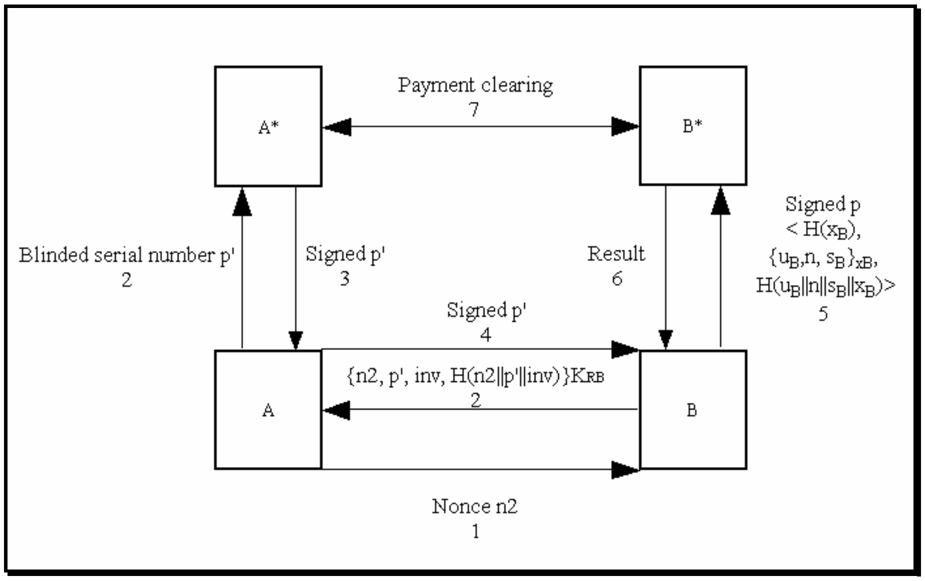


Figure 6: E-cash protocol

Following is the informal description of the e-cash protocol [6]. In this protocol e-cash is created for a specific recipient.

4.4.1 Online payment

Following steps are followed in an online payment

1. *Payer*

- generate and send nonce to payee

2. *Payee*

-generate serial number

-blind serial number

-send blinded serial number, nonce and invoice to payer with hash of all the data.

3. *Payer*

-receive blinded serial number, invoice and verify hash

-forward received blinded serial number to bank

4. *Payer’s bank*

-receive blinded serial number

-sign blinded serial number with appropriate key

-deduct corresponding amount from payer’s account

-send this currency to payer

5. *Payer*

-receive currency from bank

-send it to payee

6. *Payee*

-receive currency from payer

-verify signature of the bank

-unblind currency

-deposit in bank

7. *Payee’s bank*

-receive currency from payee

-verify signature of the payer’s bank

-send payment information to payer’s bank for clearing payment

8. *Payer’s bank*

-receive payment information from payee’s bank

-check coin against already spent or transferred coins

-clear payment

4.4.2 Online payment transfer

Following steps are followed in an online payment transfer

1. *Payer*

-generate and send nonce to payee

2. *Payee*

-generate serial number

-blind serial number

-send blinded serial number, invoice along with nonce encrypted with private key to payer.

3. *Payer*

-receive blinded serial number from payee, invoice and verify all data

-send blinded serial number and online payment information to bank

4. *Payer's bank*

-receive blinded serial number and online payment information from payer

-verify online payment information and check whether already spent or transferred

-sign the blinded serial number, deduct corresponding amount from payer's account and send signed blinded serial number to payer

5. *Payer*

-send signed blinded serial number to payee

6. *Payee*

-receive currency from payer

-verify signature of the bank

-unblind currency

-deposit in bank

7. *Payee's bank*

-receive currency from payee

-verify signature of the payer's bank

-send payment information to payer's bank for clearing payment

8. *Payer's bank*

-receive payment information from payee's bank

-check coin against already spent or transferred coins

-clear payment

4.4.3 Offline payment transfer

A user can only transfer a payment obtained from another user. New payment can not be made in offline mode. Following steps are followed in an offline payment transfer.

1. *Payer*

-send offline payment information to payee

2. *Payee*

-generate serial number

-blind serial number

-receive offline payment information

-verify offline payment information

-send blinded serial number along with offline payment information to payer's bank

3. *Payer's bank*

-receive offline payment information and blinded serial number from payee

-verify offline payment information

-check whether this payment already encashed or transferred

-sign blinded serial number from payee

-send signed blinded serial number to payee

4. *Payee*

-receive currency from payer's bank

-verify signature of the bank

-unblind currency

-deposit in bank

5. *Payee's bank*

-receive currency from payee

-verify signature of the payer's bank

-send payment information to payer's bank for clearing payment

6. *Payer's bank*

-receive payment information from payee's bank

-check currency against already spent or transferred coins

-clear payment

4.5 Generation of e-cash

We assume the following.

A - Payer,

B - payee,

A - payer's bank,*

B - payee's bank,*

aB = account number of B,

xB = secret shared by B with his bank B,*

sB = secret newly generated for each payment,

"||" *represents concatenation operation.*

4.5.1 Generating serial number

Serial number $p = H(uB\|sB)$, where $uB = H(aB\|xB\|n)$ in case user needs to transfer payment online or $uB = H(g^{(aB\|xB\|n)}(mod\ d))$ in case user needs to transfer the payment offline to another person. H represents a suitable one way hash function [8]. *g*, *d* are suitably chosen bases. Here *uB* is computed as follows. *B* generates a nonce *n*, a set of pseudorandom bits. For this class SecureRandom is used inside function getNonce().

```
        GenerateNonce genNonce = new GenerateNonce();
        byte[] nonce=genNonce.getNonce();
```

SecureRandom [31][28] class is used as follows for providing a sequence of random bits that are used as nonce or blinding factors.

```
        SecureRandom rand = new SecureRandom();
        byte b[] = new byte[10];
        rand.nextBytes(b);
```

After this *aB*, *xB* and nonce *n* are concatenated. Depending on online or offline payment transfer scenario, value $g^{(aB\|xB\|n)}(mod\ d)$ is computed or skipped. From this *uB*, for online scenario, is computed as follows.

```
        GenerateHashofG genuB = new GenerateHashofG(accNo, xB,
nonce);
        String uB = genuB.getHashofG();
```

From *uB*, serial number is generated as follows.

```
        GenerateSerialNo genSrNo = new GenerateSerialNo(uB, sB);
        BigInteger srNo = new BigInteger(genSrNo.getSerialNo());
```

BigInteger [31]class is provided for using arbitrary precision integers. It provides all the arithmetic operations which are available with traditional integer data types. It also provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation.

A BigInteger can be created in many ways. Required number can be specified as a decimal string, two’s complement representation etc. Radix of the number can also be explicitly specified. Following are the different ways in which BigIntegers can be declared and defined.

1. Using decimal string representation of the number.

```
BigInteger(String val)
```

2. Using two's complement binary representation of the number

```
BigInteger(byte[] val)
```

3. Using sign magnitude representation of the number

```
BigInteger(int signum, byte[] magnitude)
```

4. Using string representation of the number in specified radix

```
BigInteger(String val, int radix)
```

Following are some other useful functions provided by this class.

1. BigInteger modInverse(BigInteger m)

This function returns a BigInteger whose value is (this^{-1} mod m) i.e it is multiplicative inverse of this inverse with respective modulus m.

2. BigInteger modPow(BigInteger exponent, BigInteger m)

This function returns a BigInteger whose value is ($\text{this}^{\text{exponent}}$ mod m).

4.5.2 Blinding serial number

Serial number generated should be blinded to get signed from the payee's bank. Blinding the serial number is achieved by using public key of the payer's bank as follows.

```
Blinder blinder = new Blinder(pubExp, modulus);
BigInteger blindedSrNo = blinder.blind(srNo);
```

Here *pubExp* and *modulus* are the parameters of public key of bank, from which signature is to be taken. First a random blinding factor *bFactor* is created using class *SecureRandom*. The *blind* function is implemented as follows.

```
public BigInteger blind(BigInteger num)
{
blindedNum = bFactor.modPow(pubExp, modulus);
blindedNum = blindedNum.multiply(num);
blindedNum = blindedNum.mod(modulus);
return blindedNum;
}
```

After the serial number is generated and blinded it is sent to payer for getting necessary signature.

4.5.3 Sending serial number to payer

Serial number is forwarded to *A* as follows. *A* sends a nonce n_2 to *B*. Nonce is created as stated previously. *B* then sends this nonce n_2, blinded serial number *p'*, invoice *inv* and hash of all the data *H(n2||p'||inv)* to *A*, encrypted with his private key.

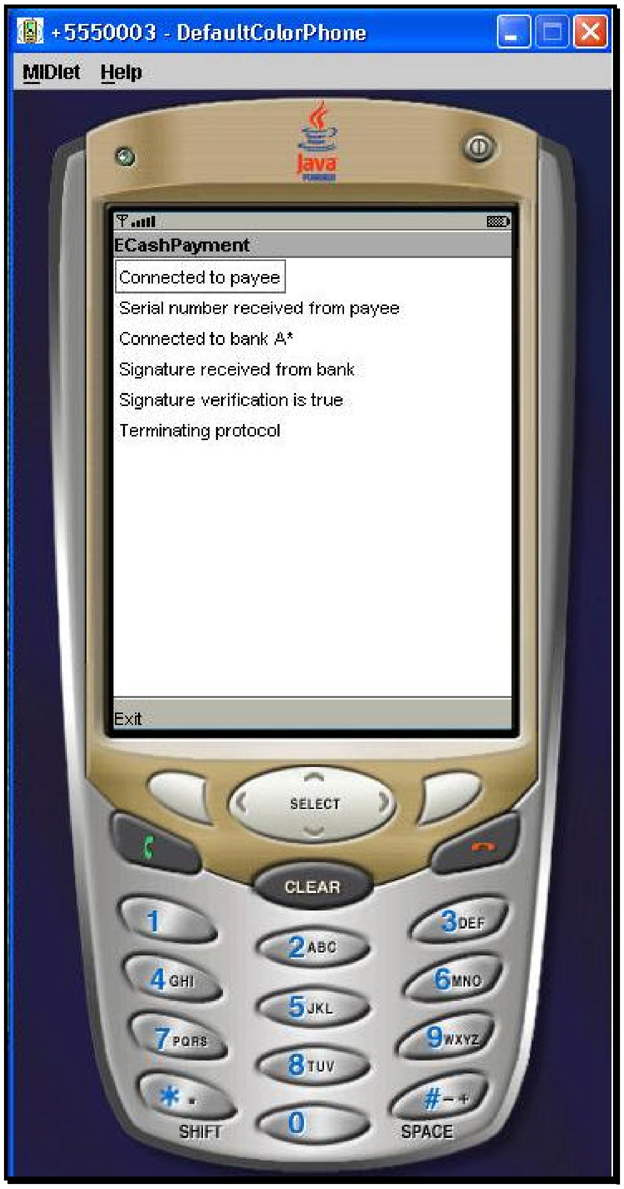
$\{n2, p', H(n2||p'||inv)\}_{KRB}$

Where K_{RB} is the private key of *B*. Hash of serial number is created using SHA512Digest [28][31] class as follows.

```
public byte[] getHash(String message)
{
    SHA512Digest digEng = new SHA512Digest();
    digest = new byte[digEng.getDigestSize()];
    digEng.update(message.getBytes(),0,message.length());
    digEng.doFinal(digest,0);
```

```
        return digest;
    }
```

A decrypts this data with public key of *B*, K_{UB}. *A* then verifies all the data and with *H(n2||p'||inv)* and take signature of his bank *A** on serial number *p'* and sends the signed serial number back to *B*. Following figure shows how the payment client software works on payer's mobile phone.



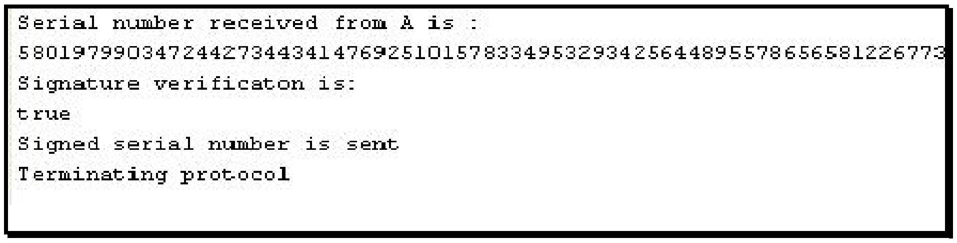
ECashPayment

Connected to payee
Serial number received from payee
Connected to bank A*
Signature received from bank
Signature verification is true
Terminating protocol

Exit

Figure 7: E-cash payer

When payer's bank receives the serial number, it signs it and sends it back to payer. Following figure shows messages generated at the payment servers of the bank when payer contacts her bank for getting signature.



```
Serial number received from A is :
58019799034724427344341476925101578334953293425644895578656581226773
Signature verificaton is:
true
Signed serial number is sent
Terminating protocol
```

Figure 8: E-cash payer's bank

4.6 Encashment of payment

For encashment of the payment, *B* needs to deposit this payment in his bank *B** by proving his identity. Along with this *B* also needs to send nonce *n* used in the creation of the cash. He sends following information to his bank.

$$\{\ H(xB),\ \{uB\|n\|sB\}_{xB},\ H(uB\|n\|sB\|xB)\ \}$$

where *H* represents a suitable one way hash [24] function. Hash is calculated as mentioned previously. Here the component $\{uB\|n\|sB\}_{xB}$ is sent encrypted by using *xB* as secret key. For this AES [24] encryption is used. It's a symmetric key cryptosystem. AES has a fixed block size of 128 bits and key size can be of 128, 192 or 256 bits. AES is implemented as follows.

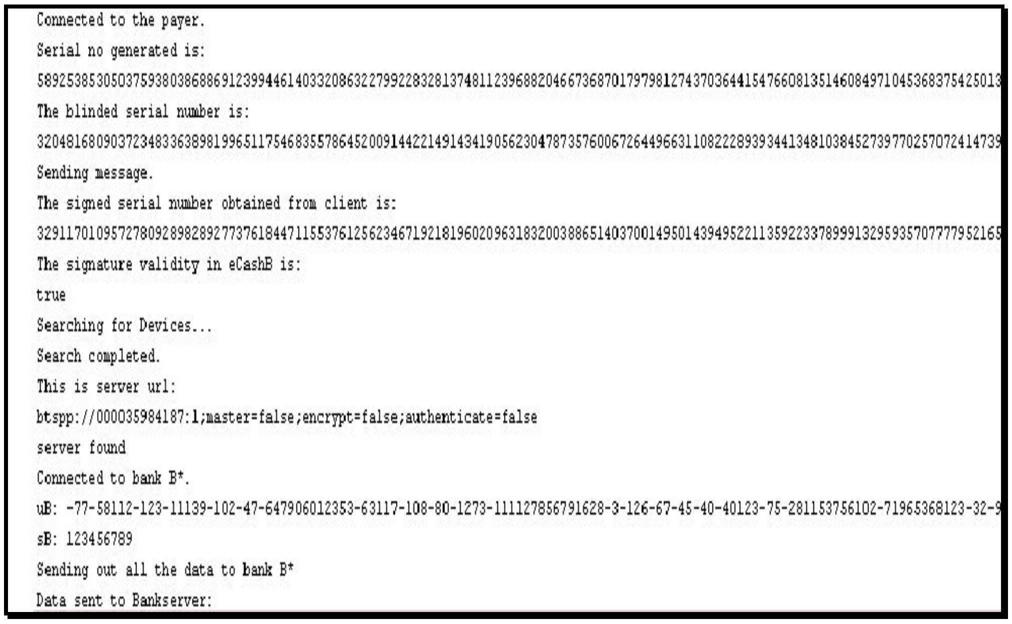
```
AESEngine encoder = new AESEngine();
String tempStr = new String();
KeyParameter keyParameter = new KeyParameter(key);
encoder.init(true, keyParameter);
byte[] newMessage = new byte[n];
encoder.processBlock(newMessage, i*16, temp, 0);
```

All information is encrypted using AES [28][31] as follows.

```
Aes aes = new Aes();
```

```
byte[] euB = aes.encrypt(uB.getBytes(), xB);
sendMessage(euB, out);
byte[] eNonce = aes.encrypt(nonce, xB);
sendMessage(eNonce, out);
byte[] esB = aes.encrypt(sB.getBytes(),xB);
sendMessage(esB, out);
```

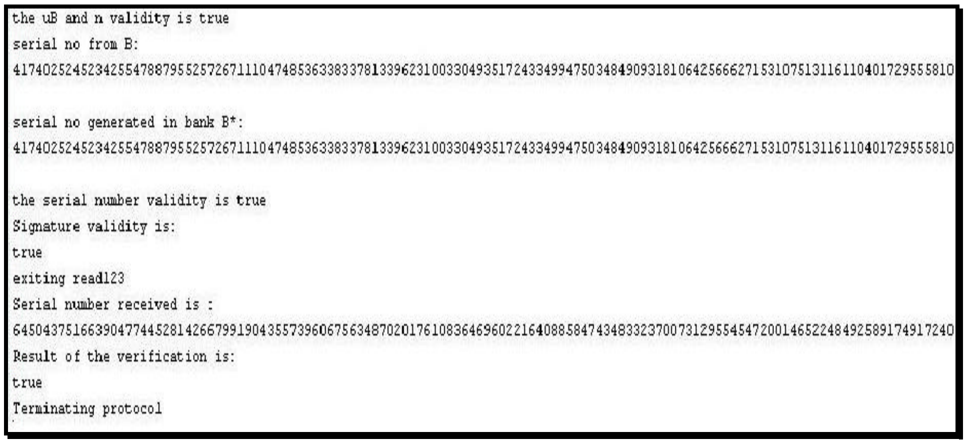
Rest of the information is sent as it is. Following figure shows the screenshot of payment server running on payee's computer.



```
Connected to the payer.
Serial no generated is:
5892538530503759380386886912399446140332086322799228328137481123968820466736870179798127437036441547660813514608497104536837542501
The blinded serial number is:
3204816809037234833638981996511754683557864520091442214914341905623047873576006726449663110822289393441348103845273977025707241473
Sending message.
The signed serial number obtained from client is:
3291170109572780928982892773761844711553761256234671921819602096318320038865140370014950143949522113592233789991329593570777795216
The signature validity in eCashB is:
true
Searching for Devices...
Search completed.
This is server url:
btspp://000035984187:1;master=false;encrypt=false;authenticate=false
server found
Connected to bank B*.
uB: -77-58112-123-11139-102-47-647906012353-63117-108-80-1273-111127856791628-3-126-67-45-40-40123-75-281153756102-71965368123-32-
sB: 123456789
Sending out all the data to bank B*
Data sent to Bankserver:
```

Figure 9: E-cash payee

After receiving payment information bank verifies that information. In implementation, records for each customer of the bank are created in a file. Each record contains name of the customer, *H*(*xB*) value and *xB* value. From *H*(*xB*) bank identifies *B*, and hence gets to know secret shared between bank and *B* i.e. *xB* . Bank then decrypts this information and gets nonce n used in creation of the cash. Then it verifies the authenticity of the cash. If bank of payer and payee is not same then the two banks communicate to clear the payment. Following figure shows messages generated at the payment servers of the bank when payee contacts its bank for depositing the payment obtained from the bank.



```
the uB and n validity is true
serial no from B:
4174025245234255478879552572671110474853633833781339623100330493517243349947503484909318106425666271531075131161104017295555810

serial no generated in bank B*:
4174025245234255478879552572671110474853633833781339623100330493517243349947503484909318106425666271531075131161104017295555810

the serial number validity is true
Signature validity is:
true
exiting read123
Serial number received is :
6450437516639047744528142667991904355739606756348702017610836469602216408858474348332370073129554547200146522484925891749172401
Result of the verification is:
true
Terminating protocol
```

Figure 10: E-cash payee's bank server

4.7 Transfer of payment

B can also transfer this payment to *C*. To do so, *C* generates a serial number $q = H(uC||sC)$. Where *uC* and *sC* are defined similar to *uB* and *sB*.

4.7.1 Online transfer

In online transfer *B* is online with his bank *B**. *B* sends *B**, *p*, payment obtained from *A*, along with necessary information for verifying *p* and blinded serial number *q*' received from *C*. Bank *B** checks the authenticity of *p* and whether *p* has already been cashed or transferred. If it is valid, then it signs *q* as bearing the value of *p*. *B* then passes on signed *q*' to *C*.

Class RSASigner is used for signing the serial number.

```
        RSASigner signer = new RSASigner(privExp, modulus);
        BigInteger signature = signer.getSignature(srNo);
```

Details of method getSignature() are as follows.

```
public BigInteger getSignature(BigInteger message)
{
        signature = message.modPow(privExp, modulus);
        return signature;
}
```

4.7.2 Offline transfer

If B is not online with his bank then, transferring only currency p with uB and sB to C is not enough. These parameters are not enough to identify B, which is needed in case B double spends. B has to pass zero knowledge proof (of aB and xB) [14] to C. B passes on a certified version of uB, denoted as UB to C. The zero knowledge proof enables C to verify the validity of the parameters of the currency he is receiving.

4.7.3 Zero knowledge proof

B passes on $M' = g^{(aB\|xB)}(mod\ d)$, $N = g^{n}(mod\ d)$ to C. M' is generated as follows.

```
GenerateMdash mdash = new GenerateMdash(accNo, new
String(xB));
```

Method getMdash() returns required value.

N is generated as follows

```
GenerateNdash ndash = new GenerateNdash(n);
```

M' is certified from his bank B^* and n is of fixed length. From this C can verify $uB = g^{(aB\|xB\|n)}(mod\ d)$ without getting to know aB or xB and thus $p = H(uB\|sB)$.

C later passes on this proof to *B**, bank of *B*, to get the required transfer signature.

Chapter 5

CONCLUSIONS AND FUTURE WORK

Contents

Chapter 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

In this research work we have discussed various electronic payment systems. Electronic cash schemes and implementation on smart phone platform was the main focus of this work. Some of popular electronic payment schemes have been studied. Different e-cash schemes have been compared. Different communication and application development technologies like J2ME and Bluetooth are also analyzed for developing electronic payment systems.

Infrastructure required for electronic payment systems has been studied. Mobile phone devices provide necessary infrastructure needed for electronic payment systems. But there are very few efficient payment schemes on smart phone platforms. In this work e-cash scheme on smart phone platform has been implemented.

Communication media for smart phones can be GPRS or SMS. But it is expensive than Internet for PCs. In this work Bluetooth is used as communication media which is free in terms of cost. But the payment scheme needs to be designed accordingly like e-cash scheme described in this work. However the protocol can be implemented with any communication media.

The protocol software has been tested on Nokia 6600 smart phone and Windows XP based PC. Integrating e-cash protocol with constrained smart phone and high end personal computers was challenging. Designing an electronic payment scheme for mobile phones without any reference implementation was challenging. In this implementation, cheaper communication standard Bluetooth is used.

5.2 Future work

1. We would like to port our e-cash protocol software on other platforms like Windows mobile and palm OS.
2. Fault tolerance of the protocol software should be improved.
3. Formal analysis of the protocol should be done using PEAR [32] tool.

REFERENCES

[1] B. Pfitzmann and M. Waidner: “Properties of Payment Systems: General Definition Sketch and Classification”, Technical Report RZ 2823, IBM Zurich Research Laboratory, May 1996.

[2] Chaum D.: “Blind Signatures for Untraceable Payments”, CRYPTO82, Plenum Press, New York, USA, pages 199-203, 1983.

[3] Chaum D. and Pedersen T. P.: “ Wallet Databases with Observers”, In CRYPTO ’92 Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, volume 658, Santa Barbara, California, pages 89–105, USA. 1993.

[4] Chaum D., Fiat A. and Naor M.: “Untraceable Electronic Cash”, CRYPTO88 Proceedings on Advances in Cryptology, Springer, Santa Barbara, California, USA, pages 319-327, 1990.

[5] Cristian Zamfir, Andrei Damian, Ionut Constandache and Valentin Cristea: “An Efficient Ecash Platform for Smart Phones”.

[6] C. R. Mandal and Chris Reade: “ A Scheme for Recipient Specific Yet Anonymous and Transferable Electronic Cash”, Proc. of WEBIST 2007, Barcelona, Spain, pages 204-209, March 3-6, 2007

[7] Ebringer T and Thorne P: “Engineering an e-cash system”, In Proceedings of 2nd International Information Security Workshop, ISW'99, Kuala Lumpur, pages 32-36, 6-7 November, 1999.

[8] Heijden and Hans van der: "Factors Affecting the Successful Introduction of Mobile Payment Systems", Vrije Universiteit, Amsterdam, 2002.

[9] H. Peterson and G. Poupard: “Efficient Scalable Fair Cash With Offline Extortion Prevention”, Lecture Notes in Computer Science 1334, 1997.

[10] L. Ferreira and R. Dahab: “ A Scheme for Analyzing Electronic Payment Systems”, In Proceedings of 14th Annual Computer Security Applications Conference, IEEE Computer Society Press, pages 137-146, December 1998.

[11] MacKie-Mason and J. K. White, “Evaluating and Selecting Digital Payment Mechanisms”, Interconnection and the Internet: selected papers from the 1996 telecommunications policy research conference, Mahwah, NJ, pages 113-134, 1997.

[12] Marten Trolin: “A Universally Composable Scheme for Electronic Cash” INDOCRYPT Lecture Notes in Computer Science 3797, pages 347-360, 2005.

[13] N. Asokan, M. Steiner and M. Waidner: “The State of the Art in Electronic Payment Systems”, IEEE Computer, pages 28–35, September 1997.

[14] Odlyzko A. M.: “Discrete Logarithms in Finite Fields and Their Cryptographic Significance”, In Theory and Application of Cryptographic Techniques, volume 209, Springer- Verlag, Berlin, pages 224-314, 1984.

[15] Okamoto T.: “An Efficient Divisible Electronic Cash Scheme”, Lecture Notes in Computer Science 963, 438–451, 1998.

[16] Rivest, Shamir, and Adleman: “A Method for Obtaining Digital signatures and public-key cryptosystems”, Communications of the ACM, V01.2 1, pages 120- 126, 1978.

[17] Shon T.W. and Swatinan P.M.C.: “"Effectiveness Criteria for Internet Payment Systems", Internet Research: Electronic Networking Applications and Policy, Vol. 8, NO. 3, pages 202-218, 1998.

[18] Anders Cervera: “Analysis of J2ME™ for Developing Mobile Payment Systems”, Master's Thesis in Information Technology, IT University of Copenhagen, 2002.

[19] Mansour Al-Meaither: “Secure Electronic Payments for Islamic Finance”, PhD Thesis, University of London, 2004.

[20] Bruce Hopkins and Ranjith Antony: “Bluetooth for Java”, Apress, 2003.

[21] Donal O’Mahony, Michael Peirce and Hitesh Tewari: “Electronic Payment Systems for E-Commerce”, Second Edition, Artech House, 2001.

[22] James Edward Keogh: “J2ME: The Complete Reference”, McGraw-Hill, 2003.

[23] Jonathan B. Knudsen: “Java Cryptography”, First Edition, Oreilly, 1998.

[24] William Stallings: “Cryptography and Network Security”, Prentice Hall, 3rd edition, 2003.

[25] Connected Limited Device Configuration (CLDC), JSR 30, JSR 139, http://java.sun.com/products/cldc.

[26] JAVA Language
http://java.sun.com

[27] Java 2 Micro Edition
http://developers.sun.com/techtopics/mobility/j2me/

[28] Java Security: Tutorial by IBM.

http://www-128.ibm.com/developerworks/library/j-midpds.html

[29] JSR-000082 Java^{TM} APIs for Bluetooth.

http://jcp.org/aboutJava/communityprocess/mrel/jsr082/index.ht
ml

[30] JSR 120: " Wireless Messaging API",

http://www.jcp.org/jsr/detail/120.jsp

[31] Lightweight Crypto APIs from Bouncy Castle Group.

http://bouncycastle.org/documentation.html

[32] PEAR: Protocol Extendable AnalyzeR – tool for testing security protocols.

http://www.dsi.unive.it/myths/PEAR

[33] SSL 3.0 specification

http://wp.netscape.com/eng/ssl3/

[34] Windows PC and mobile phone conncection through Bluetooth

http://www.benhui.net/

[35] WWW for Bluetooth

http://www.bluetooth.com

Appendix A: User manual

In this implementation mobile phones and personal computers are used to carry out payment transactions. Payment client, used to make payment, running on mobile phone interacts with payment server, used to accept payments, running on personal computer. Client part has been implemented in J2ME and server part has been implemented in J2SE.

Client side software is deployed as a JAR file. It can be transferred and installed on mobile phone in a single procedure. Mobile phone should support java with CLDC 1.0 [25] or later and MIDP 2.0. It has been tested on a Nokia 6600 mobile phone.

Server side software is deployed as JAR file. It should be installed on a personal computer having java runtime environment i.e. JDK 1.2 or later. Client and server communicate using Bluetooth network. So, mobile phone and PC should support Bluetooth.

Following is the procedure for installing Bluetooth hardware [34] and software [34] on a personal computer.

1. Install a Bluetooth dongle (H/W device) on the PC.
2. Install service pack 2 on Windows XP based PC.
3. Install Bluecove library on PC.

Mobile phone should support Bluetooth and let java application access Bluetooth.

Using e-cash application

1. Install e-cash payment client on mobile phone and e-cash payment server on PC.
2. Launch e-cash payment server on PC.
3. Launch e-cash client on mobile phone.
4. After launching e-cash client it will prompt for discovered Bluetooth devices. Select appropriate Bluetooth device (PC in this case).
5. E-cash client will display progress messages till the transaction is complete.

Communication using Bluetooth

For connecting a Windows XP based pc and a smart phone [34] following steps should be followed.

1. Install a Bluetooth dongle (H/W device) on the PC.
2. Install service pack 2 on Windows XP based PC.
3. Install Bluecove library on PC.
4. Start Bluetooth application on PC.
5. Start Bluetooth application on phone

Appendix B: Bluetooth using Java

Bluetooth [35] is a wireless technology for communication over short distances like 10 meters. It offers data transfer rate of 1Mbps. It is used commonly for battery powered devices. It enables ad hoc Personal Area Networks (PANs) to be created among a number of devices like mobile phones and PDA’s.

Bluetooth PAN [35] can be used to form Piconets and Scatternets. Piconet is analogous to client server architecture. It is the focus of this work. It supports at most seven clients to be connected to server at a time. We will discuss java Bluetooth API’s (JSR 82) [29].

A server registers itself as a Bluetooth server and processes the client requests. A client searches for various services offered by servers and connects to the matching servers for those services.
Each service is identified by a UUID which is a unique 128 bit Bluetooth identifier and service name.

A server device must be discoverable by a client. To make it discoverable following line of code can be used.

```
        LocalDevice local = LocalDevice.getLocalDevice();
        local.setDiscoverable(DiscoveryAgent.GIAC);
```

The value DiscoveryAgent.GIAC [20] (General/Unlimited Inquiry Access Code) is used to indicate that all remote devices will be able to find the device. The value DiscoveryAgent.LIAC limits the device's visibility.

Each Bluetooth service is described in a service record. The record is a set of (id, value) attributes. It can be accessed using the following line of code.

```
ServiceRecord record = local.getRecord(server);
```

The call to acceptAndOpen() makes the server block until a client connection arrives,

```
StreamConnection conn = server.acceptAndOpen();
```

Maximum number of concurrent Bluetooth connections allowed to a device can be obtained by calling LocalDevice.getProperty() [20] with a "bluetooth.connected.devices.max" argument.

```
Int maxConnections = Integer.parseInt(LocalDevice.getProperty
("bluetooth.connected.devices.max"));
```

maxConnections can vary from 1 to 7.

Device discovery

Following figure shows the stages involved in device discovery.

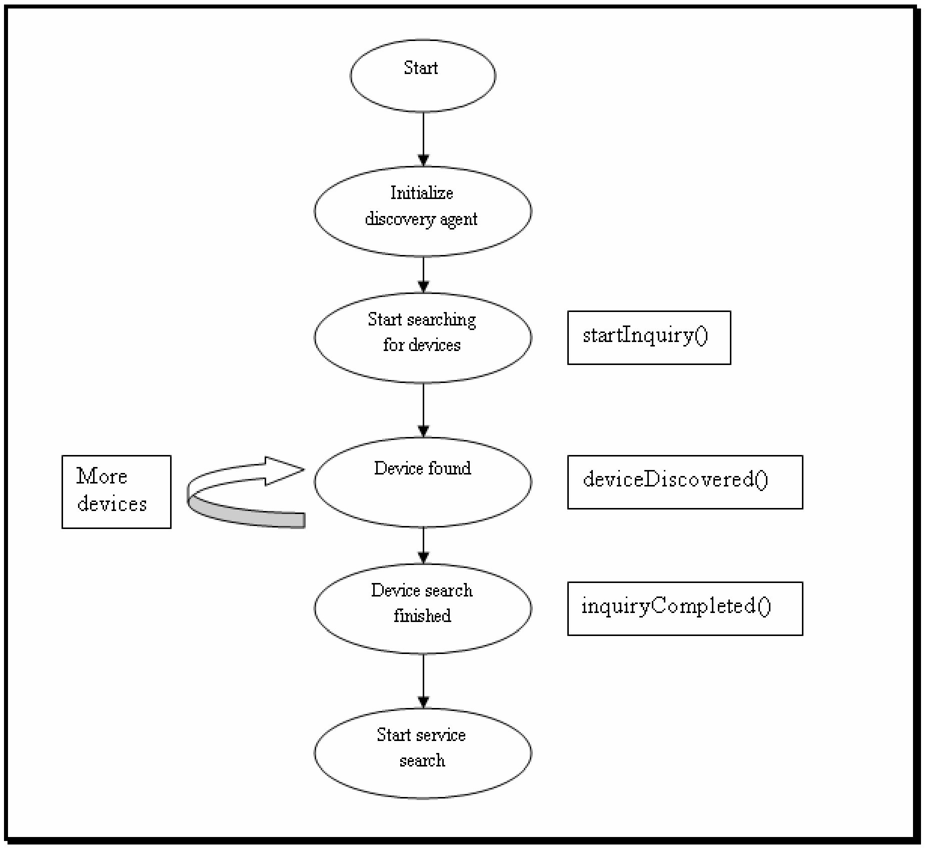


Figure 11: Device discovery

DiscoveryAgent.startInquiry() [20] function call is non-blocking, so the system indicates its progress events by calling DiscoveryListener.deviceDiscovered(),

and DiscoveryListener.inquiryCompleted().

Function DiscoveryListener.deviceDiscovered() is called whenever a device is discovered. The details of the device are checked and it is used for finding services if it is the device of our interest. When no more devices can be found, DiscoveryListener.inquiryCompleted() function is called by the system.

Service Discovery

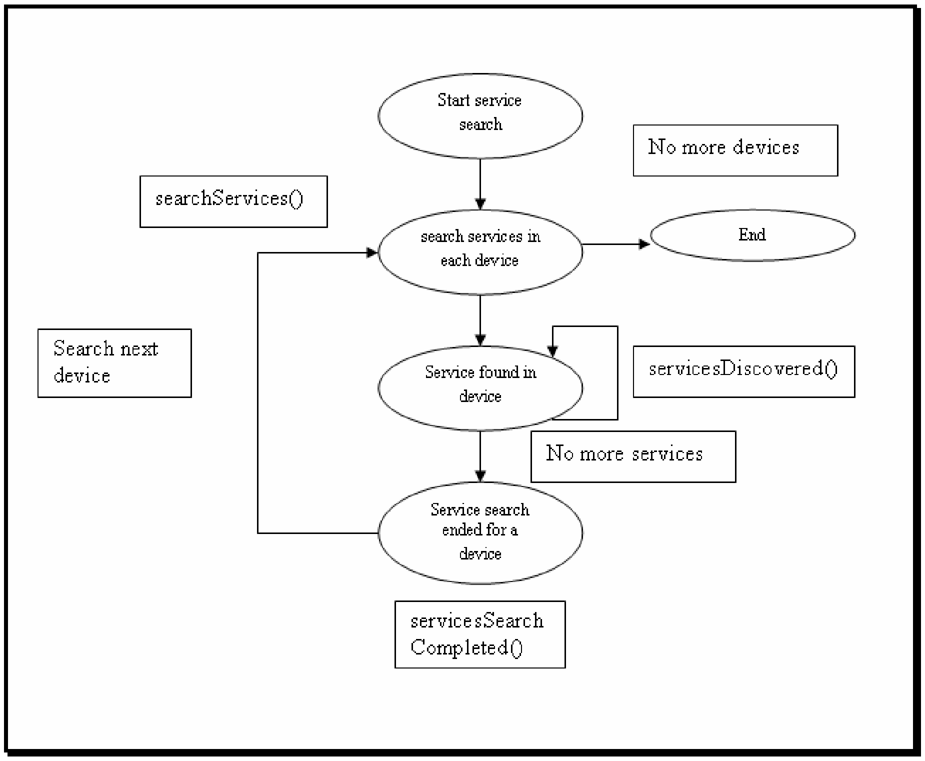


Figure 12: Service discovery

The state diagram indicates two different loops. The outer loop checks each device for services while the inner loop checks all the available service on each of the devices. Whenever a service is discovered servicesDiscovered() function is called. When no more services can be discovered servicesSearchCompleted() function is called by the system.