Cryptographically Secured Micro Payment Scheme

Tanushka Sajith Balasooriya

158203K

Degree of Master Science

Department of Computer Science & Engineering

University of Moratuwa
Sri Lanka

January 2019

Cryptographically Secured Micro Payment Scheme

Tanushka Sajith Balasooriya

158203K

Degree of Master Science

Department of Computer Science & Engineering

University of Moratuwa
Sri Lanka

January 2019

Declaration

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non - exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works.

______________________ ______________-

Tanushka Sajith Balasooriiya: Date

Approved by:

______________________ ______________-

LtCol Dr Chandana D. Gamage Date
Department of Computer Science and Engineering
University of Moratuwa

Copyright Statement

Abstract

As more and more commercial activity moves online and electronic commerce becomes the common way by which transactions are conducted, electronic payment systems will become a critical requirement for the success of many applications. The commonly used electronic payment scheme is to facilitate the online transmission of payment card data from credit cards or debit cards and to process such transactions through a payment gateway. Also, other schemes such as PayPal or ezCash provide a service where the payment is processed online using identification data specific to the service but using funds that are based on a pre-stored credit card, debit card or stored-value card system.

All these existing online payment schemes are similar in their design and operation to credit card based payments. While such schemes called macro payment systems are appropriate for transactions with a relatively high value compared to the service charges of the online payment providers, there are many e-commerce applications that are being developed for which such credit card style payment schemes are inappropriate.

As a solution to this problem, research has been conducted on an area called micro payment systems. The design of micro payment systems need to be radically different from the macro payment schemes as properties such as online real time availability of all participants, use of public cryptography schemes, availability of high computation power, etc that is common for macro payment systems are not desirable in micro payment systems. Taking into consideration the context in which micro payment schemes operate, where a transaction value is very low, micro payment systems need to be designed with a careful trade-off between reliability and cost of implementation.

This proposed research is intended to study the properties of existing micro payment systems, evaluate the strengths and weaknesses of those schemes, and prepare a model for a micro payment scheme with only the most essential properties. The research further envisages the selection of cryptographic mechanisms and development of protocols to implement this micro payment model.

Acknowledgements

First of all I would like to thank my supervisor Dr. Chandana Gamage whose encouragement, guidance, support, and criticism from start to the very end, allowed me to understand the objectives and challenges of a master degree thesis. I would also like to thank Dr. Indika Perera (Course Cordinator) for extensive advice, helpful feedback, and constant support.

Furthermore, my special thanks go to Dr. Shantha Fernando who provided an excellent, supporting, innovative, and inspiring environment in which it was a pleasure to create this thesis.

Finally, words alone cannot express the thanks I owe Mr. Premaratne my father, Mrs. Amara Weerasinghe my mother, Ms. Jeewantha Sandamalee my sister and Ms. Lakmi Bandara my loving girl friend for all the encouragement extended.

Abbreviations

CA - Certificate Authority
CAFE - Conditional Access for Europe
CSP - Communicating Sequential Processes
DoS - Denial-of-Service
FSTC - Financial Services Technology Consortium
ISP - Internet Service Provider
PAT - Protocol Analysis Toolkit
PKI - Public-key Infrastructure
POC - Proof of Concept
RSA - RivestShamirAdleman
SET - Secure Electronic Transaction
GSM - Global System for Mobile communication
NFC - Near Field Communication
SMS - Short Messaging Service
EMV - Europay MasterCard Visa

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Concept of Micro Payment

An electronic payment system which is designed to allow frequent payments of small amount as small as a fraction of a cent is called a micro payment scheme. Micro payment schemes should have to minimize the computation and communication engagement as much as possible to keep the transaction cost very low and scheme to be efficient.

Due to the major concerns over efficiency, the security requirements are not much tight and it is acceptable as long as the amount involved with transactions are small. Micro payment schemes uses lightweight cryptography to provide offline authentication and payment verification.

The security concerns over micro payment schemes are less due to the cost for a fraudulent transaction is costly than the actual gain from it. The majority of existing micro payment schemes were designed to pay for consultation services, usage of an online database system, refer financial Web pages, listen or watch videos or songs, or play an online game. Micro payment scheme can use to pay for voice and data transport services as well as the quality enhancement of the services provided online.

Micro payment schemes can be classified with varies scope segments. When considering the model of the scheme, notational model and the token model are two type of micro payment schemes. In the model of notational model of micro payment schemes, the payment message is involved with the users transaction which includes the value of the payment and the payment orders. Some of existing micro payment schemes on the notational model are Millicent [20], Micro-iKP [13] and NetBill [31].

If the micro payment transactions exchange tokens instead of payment messages, it is classified as token model micro payment schemes. The token can be a representation of coins or bank notes. The PayWord [29] and MicroMint [29] are micro payment schemes of token model. This type of micro payment schemes are efficient for frequent small payments.

Due to the requirement of efficient process with low transaction costs, a micro payment scheme is required without unnecessary communications between external parties and expensive cryptography mechanisms.

1.2 How Micro Payment Differentiate with Macro Payments

Many ecommerce systems are used Macro payment systems [10] because of the proved security features and other necessary functions which should be included in every ecommerce transaction. While engaging with macro payment systems the customer has to pay for the service or good when he purchases it or before purchases it. That will be mostly conducted by debiting credit card, real time bank transfers or digital cash. Currently these systems are used complex public key encryption systems and also the servers should communicate to request and confirm the transactions. This model is suitable for paying considerable amount.

Today in Internet, most publishers are selling news and it can be separated with section by section like newspaper [7] which is allowed to read article by article with paying if customers wish to read it. Therefore, having on-the-go paying systems are mostly looking for micro payment models which are having very low transaction cost. If the transaction cost is more than the amount to pay, the readers will not much engage with that product any more.

Generally mobile devices were in the category of lower power, small storage capability, and lower computational capabilities devices compared to other computing devices. Therefore, those devices were not efficiently performed high computational operations such as public-key encryptions. But recent mobile devices are with high computational power and storage capabilities.

Also the communication cost involve with wireless networks is higher than wired connections. Therefore, mobile payments schemes with existing payment protocols are not much acceptable for micro payments by many users.

There are several payment protocols which are suitable and designed for the fixed networks. Because of the use of public-key infrastructure, it is not efficiently applicable to wireless networks. With these models, client and merchant required to connect with the bank or broker and the public key certificate authority every time.

According to the design of several existing payment protocols, during a transaction, merchant and client has to be verified by a Certificate Authority (CA) which results in additional communication cost.

1.3 General Payment Gateway

A general payment transaction model which based on account is based on 4 parties. Those parties are client, merchant, clients financial institution which is also called as issuer and merchants financial institution which is also called as acquirer [5]. Figure 1 shows the general model for payment transaction [17].

The payment gateway is an additional party for above mentioned four parties but it works like a postman between issuers and acquirers at banking network side. It also acts as a medium between merchants and clients at the Internet side for purpose of clearing [23].

According to the above general payment model, payment is made by a client to a merchant. The client makes the value subtraction on behalf of an issuer from client's account and the merchant makes the value claim on behalf of an acquirer to transfer money to the merchants account [17].

Many electronic payment systems have been developed. In general, Considering the amount of the payment value, electronic payment schemes are

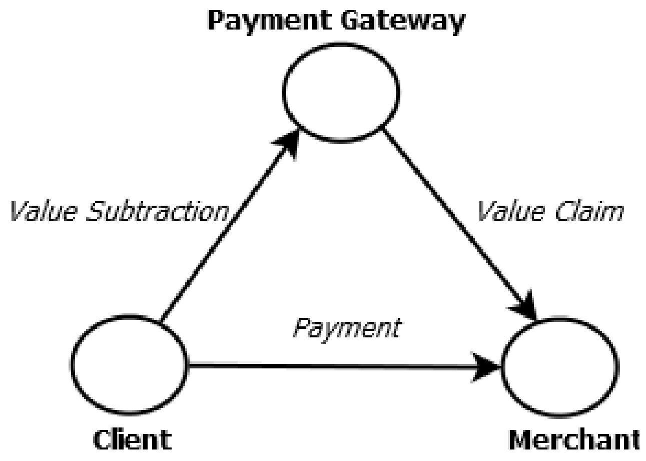


Figure 1.1: General Payment Model

categorised into macro payments and micro payments.

There are several payment schemes and security requirements handled by them are varies with use of the commercial application. Therefor the existence of several type of micro payment schemes not suitable for any m-commerce apllication. This section review the existing payment schemes and techniques with the cryptographic algorithms which are using to model and develop the payment schemes.

Electronic payments schemes which are aimed to allow payments between one rupee to several thousands of rupee known as macro payment schemes. If the payments are larger than the category under macro payments are use the private network within banks. Based on the payment method, the macro payments can be classified again. Those are credit cards based, electronic, and digital cash.

Because of the amount of transaction, macro payment schemes use complex encryption algorithms and each payment need to be verified with payment authorization servers to request or confirm.

Payment schemes which are based on credit cards are online and postpaid schemes, SET and iKP are examples for credit card based payment systems. iKP is flexible to implement with different type of security requirements depending on the application requirement.

There are several payment systems depends on cash-like payment. Some of them are e-Cash and Conditional Access for Europe (CAF). A well knows cheque is considered as a signed order to transfer funds from payer's bank account to payee's bank account. In the aspect of electronic cheque scheme, the signing part is happening digitally and digitally signed message is transmitted through the network.

There are few examples of payment schemes which are using digitally signed cheque concept and famous parties are Netcheque and Financial Services Technology Consortium. This type of payment schemes are having heavy communication and computation cost to process a single payment although the protocol is providing high-level security. Because of the cost for single transaction is higher than the transaction amount, these schemes are not suitable for micro payment.

Electronics Commerce sites are using electronic payment which are based on paperless monetary transactions. Electronic payment has reduce the paper works and because of that, the business processing capacity increased. Transaction costs, labor cost are getting decreased as the most of the paper works are happening automatically. Business organizations can expand globally as the payment flexibility and the less time consuming tasks than processing payment methods manually [18].

Transactions in electronic commerce applications and payments can occur without any prior identified interpersonal relationships or prior human contact. This lack of mutual trust generates some sort of a security threat. A well accepted definition of e-commerce is "the sharing of business information, maintaining business relationships and conducting business transactions by the means of telecommunication networks" [27].

It is generally defined that "the electronic payments phase is the ultimate test of security and trust in e-business environment and Adoption of payment mechanisms and electronic money as other forms of payment depends upon trust in the security and reliability of the system and control of the particular transaction" [27].

1.4 Cryptography

Cryptography which is the study of techniques for communicating securely when there are involvements with third parties, is about innovating, developing and analyzing protocols that eliminate from reading readable clear message for third parties and/or the public. Cryptography is also consists with major aspects in information security such as authentication, integrity, confidentiality, and non-repudiation. Modern cryptography based on the advanced concepts of mathematics, electrical engineering, and computer science. There are several applications which uses cryptography in modern world including computer passwords, ATM cards, and electronic commerce [16].

The way of usage of Cryptography before to the modern days was effectively synonymous with encryption, the transformation of information to unreadable state from a readable state.

Modern cryptography techniques are based on computer science and mathematical concepts. Most of the algorithms which uses in cryptography are designed with assuming that computational hardness of solving and assuming that such algorithms are hard to break or solve with real world instruments by any third parties without knowing necessary parameters. It can be possible to break such a algorithm in theoretical perspective, but it is almost impossible to do so by any known practical means [16].

These algorithms are therefore known as computationally secured algorithms. There exist information-theoretically secure algorithms that provably cannot be broken even with unlimited computing power. But those schemes which are also theoretically secured algorithms are difficult to develop [16].

1.5 Classic Cryptography

Since most of people could not read and write, the earliest forms of secret massage communication required little more than writing implements. When the people becomes more literacy, it is required to have actual cryptography. One of the main classical cipher type is transposition ciphers. It rearrange the order of letters in the communicating message. Other main classical cipher type is substitution ciphers. What is does is it systematically replace the letters or groups of letters of communicating message with other letters or groups of letters which are defined or agreed mutually by both parties.

The Caesar cipher was the earliest substitution cipher. It replaces each letter with some fixed number of steps or positions in the alphabet. As the information from Suetonius, it is identified that Julius Caesar used the fixed number as three when communicating with Caesar cipher to his generals.

1.6 Modern Cryptography

Although classical cipher can use only with the written language text, modern cryptography can be used to generate cipher from any format of data which are possible to present in binary format. With the improvement of the digital instruments and the computers, it generates ciphers which are having more complex algorithms.

1.6.1 Symmetric-key Cryptography

Symmetric-key cryptography uses a shared key which has already shared with both parties. There are two types of symmetric-key cryptography based on the implementation. Those are known as block ciphers or stream ciphers. The block cipher uses blocks of the input plain text. The stream cipher uses individual characters of the input plain text.

Stream ciphers uses any kind of lengthy stream which is heavily defers from the 'block' concept. This concept create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by-character, somewhat like the one-time pad. In a stream cipher, the output stream is created based on a hidden internal state that changes as the cipher operates.

The pseudorandom keystream is typically generated serially from a random seed value using digital shift registers. The seed value serves as the cryptographic key for decrypting the ciphertext stream.

Stream ciphers typically execute at a higher speed than block ciphers and have lower hardware complexity. However, stream ciphers can be susceptible to serious security problems if used incorrectly [16].

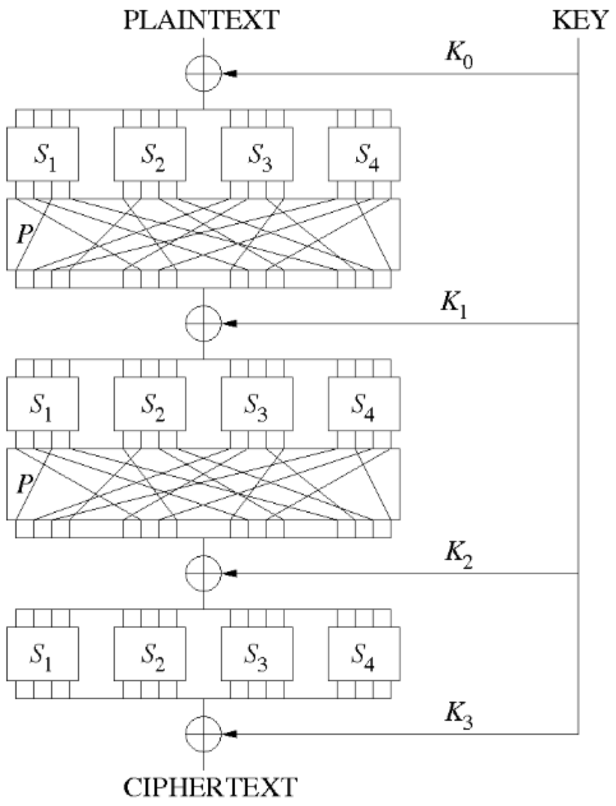


Figure 1.2: Iterated Block Cipher Diagram

1.6.2 Asymmetric Cryptography

Asymmetric cryptography which also known as public key cryptography is a most popular concept in cryptographic system. Asymmetric cryptography uses pairs of keys. Those are public keys and private keys. Public keys are known widely for everyone. But the private keys are known and accessible only to the owner of the key pairs.

Using the public key of the receiver which is known by everyone, sender encrypt the plain text and then sends to the receiver and only the receiver can decrypt the encrypted message as that was encrypted using public key of private key which own by the receiver.

There should be a strong algorithm generating a public key and private key pair as the strength of the cryptography depends on the how hard to solve the private key from the related public key. The security of the cryptography is based on the how much securely stored the private key without exposing it to publicly. Public key can be expose to public without any concern if the above strength and security can be provided [30].

Computational complexity of the asymmetric encryption is higher than others encryption and therefor it is normally used only for small set of blocks of data. Sometimes it uses to transfer the shared key which uses in symmetric encryption. This shared key is then used to encrypt and decrypt the long messages of larger size of data.

Generation of key pair using RSA:

- Generate the modulus (n)
  - Select two large primes, p and q.
  - Calculate $n = p * q$. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.
- Find Derived Number (e)
  - Number e must be greater than 1 and less than $(p-1)(q-1)$.
  - There must be no common factor for e and $(p-1)(q-1)$ except for 1. In other words two numbers e and $(p-1)(q-1)$ are coprime.
- Form the public key
  - The pair of numbers (n, e) form the RSA public key and is made public.
  - Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n.
- Generate the private key
  - Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
  - Number d is the inverse of e modulo $(p-1)(q-1)$. This means that d is the number less than $(p-1)(q-1)$ such that when multiplied by e, it is equal to 1 modulo $(p-1)(q-1)$

- Encryption
  - Suppose the sender wish to send some text message, P to someone whose public key is (n, e).
  - Ciphertext C is equal to the plaintext P multiplied by itself e times and then reduced modulo n.
- Decryption
  - Receiver raises C to the power of his private key d. The result modulo n will be the plaintext P.

1.7 Research Problem

The micropayments which are using transaction value as less than 1 rupee are becoming popular payment service on the web. The existing subscription models are demanding higher upfront payment and therefor any replacement model which suitable for micropayments become more popular. Today, there are several systems which allow users to pay an upfront amount for some kind of products, such as an access to paid premium content [25].

The micropayment schemes aim is to simplify the process of lower electronic payments but still not addressed the every aspects of requirements. Becuase of the implementation issues the industry is not catch the existing micropayment schemes [11]. "Micropayment schemes need to make their systems fully reliable, secure, and easy to use. Not only is the billing method a technical challenge, but so is the user interface" [26].

The user friendliness of the micropayment schemes should have to be very higher as most of the existing schemes have not much popular due to concerns over Downloading mobile application or desktop software, authenticating and verifying bank accounts, and monitoring commissions and charges continuously.

Economic and social impact is also a major concern over popularity of a payment schemes and therefor implementation aspect as well as social and economic aspects should have to concern when new payment scheme proposed.

By examining the challenges in social, economic, and technological in current micropayment schemes, there should be a Micro payment system with highly secured and less transaction processing cost [24].

Most of the existing online payment schemes are similar in their design and operation to credit card based payments. Those systems are called macro payments as those are processing relatively high transaction values compared to commission value. But such credit card style payment schemes are inappropriate for many e-commerce applications which are having very low transaction value called as micro payments.

1.8 Methadology

As a solution to this problem, research has been conducted on an area called micro payment systems. The design of micro payment systems need to be radically different from the macro payment schemes as properties such as online real time availability of all participants, use of public cryptography schemes, availability of high computation power, etc that is common for macro payment systems are not desirable in micro payment systems. Taking into consideration the context in which micro payment schemes operate, where a transaction value is very low, micro payment systems need to be designed with a careful trade-off between reliability and cost of implementation.

Our approach is to build Cryptographically secured micro payment scheme which uses Smart phones. The solution should consist with less transaction cost and offline processing with security.

1.9 Thesis Outline

In this introduction chapter, the background to the research problem including an explanation of the concept of micro payments and how it differs from conventional payment schemes was given. Furthermore, the cryptographic background to implementation of micro payments, including public key cryptography techniques were described. The chapter was concluded with a formal definition of the research problem and the intended research methodology for the proposed study.

In chapter 2, the historical developments and current state-of-the-art in micro payments are reviewed through a detailed literature survey. The discussion includes a detailed analysis of the cost structure for micro payments in terms of computational, communication, etc as well as an examination of the well known micro payment schemes in existence. The chapter concludes with an overview of the outstanding challenges in the design and implementation of micro payment schemes.

In chapter 3, the proposed solution which is for micro payment scheme using digital signature is proposed with analysis of properties. Furthermore the analysis in this chapter includes the drawbacks of the use of digital signature also. This chapter provides proposed scheme overview diagram and detailed descriptions for each steps and phases of the scheme also.

In chapter 4, micro payment scheme using ring signature is proposed by considering the drawbacks of the use of digital signature. It also analyse the properties and how it overcomes the drawbacks of micro payment scheme using standard digital signature. This chapter provides improved version of proposed scheme overview diagram and detailed descriptions for each steps and phases of the scheme also.

In chapter 5, analysis of protocol is given. This chapter models the client-side using CSP. The model of examined in this chapter covers the handling of e-coins and the parts of the protocol that deals with registration and redemption is not examined. This chapter is to prove that the protocol is non-blocking and deadlock free protocol.

In chapter 6, the Proof-of-Concept and implementation of the solution is given. The mobile application that would run a client on a smart phone device for both a customer and a merchant with Bluetooth communication for exchanging messages during a transaction is proved. The correctness of the cryptographic hash function is assumed and not proved within the chapter. The chapter is to prove that the solution is possible to implement and usable practically.

In chapter 7, concluding remarks are made on the research along with summaries on guidelines for implementing micro payment schemes and the properties of the micro payment scheme developed as part of this research study.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews several of existing micro payment schemes and contains mechanisms that can implemented on both distributed networks as well as peer-to-peer networks.

At the end of this chapter a comparison analysis of the schemes is provided as a summary.

2.2 History of Micro Payment System

Digital goods are expensive to produce but the reproducing cost is almost zero compared to production cost. Digital goods are not restricted to single customer and therefor multiple partied can consume same resources same time.

This approach applicable to digital services which includes stock quotes, newspaper in digital format and interactive services like massive multiplayer online games.

Micropayment systems usage increased for digital content in early 1960s when Ted Nelson create web hyperlinks which works on both directions. Then this made possibility to use for paying digital content they access also. Tim-Berners-Lee and Marc Andreessen propose to include micropayments to web protocol level in the late 1990s. But that proposed solution was discouraged by banking regulations at that time [6].

There have been two generations of innovations in micropayment systems [6] [19]. First in mid of 1990s, inspired by the electronic cash movement. Because of the low processing power and dial-up internet infrastructure, the main design goal was to minimize the processing and communication cost for transactions.

Popular strategy in those systems was prepaid or based on account approach which used by systems like NetBill, CyberCoin and Mini-Pay. A centralized ledger which maintains by broker prevents frauds like double-spending.

The Crypto-token approach was the second strategy which used by systems like MicroMint and PayWord. Those systems generate tokens cryptographically which are infeasible to counterfeit. Some of these systems were operated with popular known brands like Visa and IBM but because of variety reasons such as poor usability, requirement of trusted hardware and high latency the systems were not much succeed.

The second generation rises around 2000. Most popular strategy was account-based approach which use systems like Paypal and ClickandBuy. First users have transfer money into account and then they can spend for several merchants. Most of the micropayment systems on this strategy have survived to the present day because of the properties such as more usability and low latency.

Several organizations including World Wide Web Consortium, the PayCircle Consortium and Secure Mobile Payment Services project works for standardizing the micropayment protocols.

2.3 Properties and Requirements

A mobile commerce system needs to provide secure services and required properties to its participating entities so that business can be conducted successfully in electronic form.

- Authentication. The both parties who involve to a transaction should have to provide and confirm the identity each other. Using this confirmation, it is possible to remove the unauthorised parties being act as a legitimate party.

- Data confidentiality/secrecy. Only the sender and receiver should have a proper way to view a message in clear text and this will make sure that no other third parties can view the massage in readable format although captured while transmitting the messages.

- Data integrity. Only the sender and receiver should have a proper way to view a message in clear text and this will make sure that no other third parties added or removed content from the transmitted message. Use of digital signature provide the possibility to achieve this security property.

- Non-repudiation. There should be a proper way to identify the sender of the message as no one should be able to decline the already sent message. Also this should provide a way to protect other parties to send message by acting as a different authorised party.

- Availability. The payment schemes should be available anytime anywhere as everything in the world becomes online real time. Therefor when ever a legitimate party want to make a payment, the payment scheme should have the ability to provide an uninterrupted service.

- Processing fees. Processing fees consists of infrastructure and clearing costs which includes computation, equipment, communication, accounting and storage cost. There are cognitive or mental transaction costs as economists describes [6].

- Anonymity. Anonymity refers to the exposure of identity of the payer or customer to the broker or merchant due to use of payment gateway. Physical cash achieves total anonymity but with credit card, both bank and merchant privy to the payers identity and order details. Some commercial payment systems provide total anonymity, and some offers partial anonymity. Strong anonymity can become a mechanism for crime and money laundering also. There for most of the systems design goal is not anonymity from the bank [6]. Another design of anonymity is revocable anonymity which obtain customer privacy in case of dispute.

- Validation. A system required to real time validate the transaction by online communication with broker or with offline process.

- Transferability. Transferring funds using payment system among customers and merchants denotes transferability.

- Payment mode. Some systems are prepaid which should input funds prior to the transaction process and some are postpaid which will be track and debit total transaction cost at the end of billing cycle.

- Usability. Ease of use of the payment system.

- Scalability. The ability to increase customer base and transaction volumes without affecting payment system reliability and performance.

- Security. Systems resilience to fraud with ability to prevent counterfeiting and double spending. Most commercial systems maintain each user balances and explicitly authorize every transaction. Some systems use distributed and highly synchronized using a ledger called blockchain. Some systems use cryptography to generate crypto-tokens which are too complex and expensive to counterfeit.

The requirements for mobile commerce payment methods and security features are:

1. Integrity.
2. Confidentiality.
3. Authorization.
4. Availability.
5. Authentication.
6. Interoperability with existing systems.
7. Non-repudiation.
8. Efficiency and user-friendliness.
9. Acceptability of the the current systems with reduced cost.

2.4 Transaction Costs of Micro Payments

The main purpose of micropayment systems is the purchase of items which are under category of low value. With the growth of mobile network technology, micropayment systems have considered as popular and useful technology.

Micropayment systems are used with very low value goods and therefor it is required to have a significantly negligible transaction cost for each transaction without losing a performance of the system.

This transaction cost should be minimal and to make this cost minimal, there are several factors to be considered.

2.4.1 Fixed Technical Costs

When there is a payment system there are costs for the software (development, purchasing, and maintenance), hardware (purchasing and maintenance), installation procedures, and maintenance of accounts (account openings, balance enquiries, amount transfers, etc)if available with the system [2].

2.4.2 Storage Costs

Customers can challenged the transactions and auditors may recall the transaction logs anytime within some sort of time period. Therefore it is necessary to keep transaction records for required length of period set by the law. There can be lot's of small amount transactions in micropayment systems and therefore all the entities participating to the transaction will have to keep logs and so the storage is getting increased.

However, because of the releases of law requirements for micropayment, it is possible to reduce the storage requirements and keep history records archived using off-line media [2].

2.4.3 Computational Costs

When transaction requires a complex computations, it takes CPU time and also delay the wait time for customer which affect mostly for user friendliness also. Both side which participate to transaction need to sign something and should verify each other also. This is almost equivalent to contract with both sides. Having less signatures and verification can speed up the transaction time and reduce the computational cost.

In terms of processing of transaction, another costly phase is settlement phase. Most of the micropayment systems do the settlement phase independent of transaction phase. Instead of settling each transaction actual currency, collect several micro transaction amounts and settle once a time duration [2].

2.4.4 Communication Costs

Most of the transactions delay happens with communication bottleneck. Micropayment systems mostly do not like to use public key cryptography because of this communication bottleneck as it requires to connect with third parties. The number of messages communicating between every parties involving to transaction makes the systems not only complex but also increase the speed and cost also. This is also impact to delay the responses of inquiries which are failed during the transactions. It is also increase the possibility to break the transaction when there are several messages required to communicate and more third parties involvements required.

2.4.5 Administrative Cost

One of the most costly factor in the administrative cost is billing cost. Use of paper materials for billing is inappropriate as transaction value is almost fraction of cents. There should be a electronic billing system and verification process for parties involved to make the billing cost to be low.

It is required to have a profit for the administrative party who introduce the commercial micropayment system. This also should be profitable to merchant and easy to use. Then only most of the users become merchants and involve with micropayment system to sell goods and services [2].

2.5 Micro Payment Schemes

2.5.1 MILLICENT

Millicent went live on 1999 in Japan. This micropayment scheme is not going to be on either on-line or off-line category. The new type of currency called Scrip which is digital money introduced by the Millicent. Scrip which is relevant to specific vendor has a value as traditional cash. Scrip also consist with expiry dates, the information regarding the specific vendor and a particular serial number holds a particular value [20].

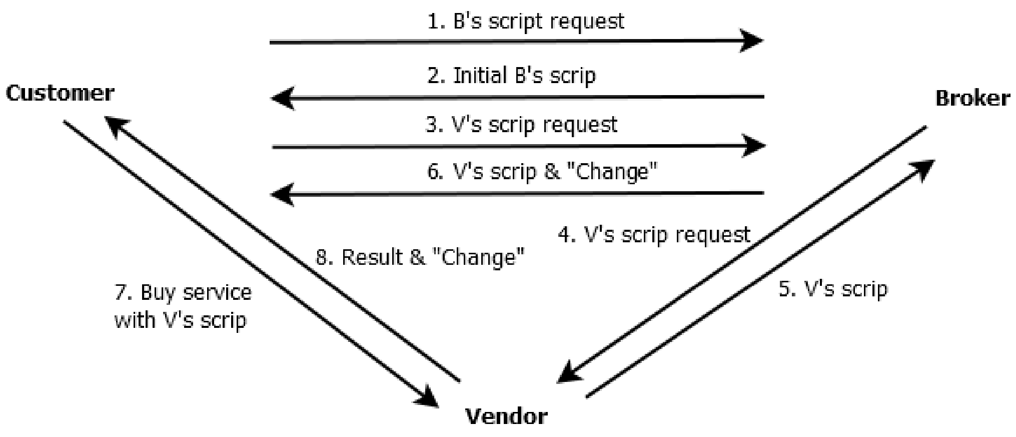


Figure 2.1: Millicent Payment Model

Broker scrip request: At start of the day, the customer buys the broker scrip. At that stage the broker returns associated secret with initial broker scrip.

Vendor scrip request: There are three models of customer requests for vendor scrip and paying for them with broker scrip from the broker.

- Scrip warehouse model: the broker asks the scrip from the vendor directly.
- Licensed scrip production model: The vendor gives the permission to the broker to generate vendor scrip that the vendor can validate and accept if the broker requests more scrip from specific vendor.

- Multiple brokers model: customers broker needs to contact vendors broker to buy the scrip.

Transaction: Using vendor scrip, the customer can buy the goods or services. After buying the goods or services, the vendor should have to return the information goods and "change". The customer has the possibility to continue more purchases using change with this same particular vendor.

There are no public key cryptography and customer can repeatedly use scrip for same vendor. Because of the distributed approach which uses Millicent, it prevents the double spending and also validate the scrip. Also this validation and prevention can be done without contacting the broker online.

One of the major issue of Millicent is the broker who generated the scrip should be available on-line when the customer wants to purchase goods or services with a new vendor [20].

The other drawbacks are,

- The customer should have to have the possibility to interact with the broker almost anytime in order to make payments.
- The vendor scrip is generated for a specific vendor and it has no value when use with another vendor.
- The transactions with different vendors having different brokers can be very complex.

2.5.2 MPAY

The system called Mpay is introduced by IBM, 1999. Current third party value added services from phone networks are similar process of this system. Most of the inexpensive information/services uses this model due to the suitability of the system for them.

The customer interacts with his issuer and the vendor interact with his acquire. At the end, the issuer and the acquire settles the accounts [29].

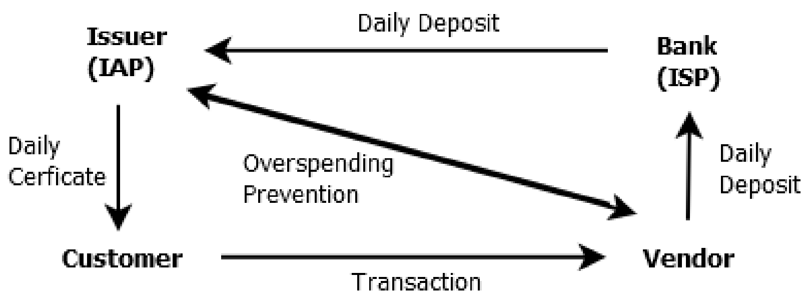


Figure 2.2: MPay Payment Model

Daily certificate request: The customer should connect every day to his issuer to receive a daily certificate. This certificate signed by the issuer whose public key is known by all vendors. The certificate inform that the customer has an account and the recommended offline limit within daily purchases.

Transaction (online): There is a tag in customers browser to click. The system encodes the cost, daily certificate and other necessary information and sends to the server. This allow the customer to send the payment at same time as the query for the vendor.

The vendor should have to verify the issuer's signature which is on the certificate of issuer. The vendor should verifies not only the signature but the daily limit also. The vendor will responds to the request immediately if the signature verification is success and the daily limit is not exceeded. If the daily limit is exceeded, the vendor should have to generate an extra spend request.

Daily deposit: The vendor creates a single signed deposit message after fixed duration period which includes all the customers payment orders and sends to the bank. The bank does the same thing. The bank creates a deposit message by sorting the each customers payment orders from all the vendors and then sends to the issuer of the each customers.

Daily process: The customer should have to contact the issuer at the beginning of the next day or at the end of every day for their daily process. Issuer of the customer and the customer compare last cutoff to present cutoff records and clear matching records. Then insert new summarized and signed document of purchase and the balance.

Mpay is having offline capacity because of the daily certificate issued by the issuer. Because of the use of one or no public key operations the transaction cost is very low. The customer can use it easily.

Major drawback of the system is customer can leave the payment to issuer after purchasing something online. But the issuer can protect itself by having a prepaid deposit. Other one is issuer can aware about the purchase profile of their customers [29].

2.5.3 PAYWORD

This protocol is introduced by Ron rivest in 1993. Hash functions which are faster than public key operations are used in this protocol. Customers are creating paywords and used to purchase from vendors. Vendors verify them with broker [29].

PayWord certificate request: The customer should have to create an account under broker at the beginning of the transaction. Broker is the party who issues a digitally signed PayWord certificate. This payword certificate includes identity and public key of the customer. Use of this payword certificate is to establish the trust for vendors that paid paywords are redeemable by the broker.

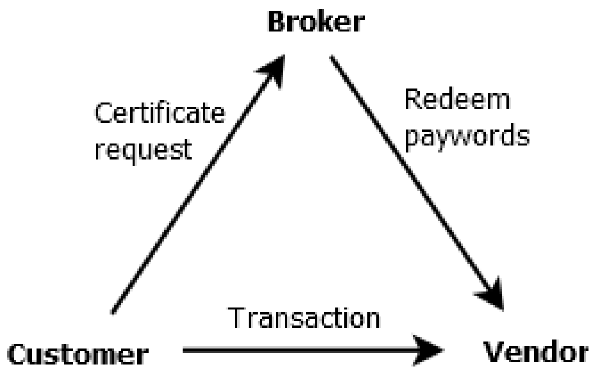


Figure 2.3: Payword Payment Model

Transaction: Before the very first purchase transaction of the day, customer randomly select a seed for the payword w_n. Payword chain is generated by the customer by repeatedly hashing $w_n : w(i-1) = h(w_i)$, where $i = 1, , n$.

It is required to send the w_0, the seed of the payword chain and the certificate to the vendor to establish the trust and to show the customer's intentions of spending paywords. If the cy=ustomer wants to make a payment worth m cents payment, the customer sends w_1 through w_m. In this notation the m is the number of the paywords.

The vendor receives the w_m and then verify this chain by hashing w_m m times until he collect the w_0. The vendor can verify the received paywords quickly using this method. After the verification the vendor delivers the information goods or the services to the customer.

Redeeming: Every day after the completion of transactions, the vendor sends each of the customers highest payword spent to the broker. The broker should have to verify the paywords. Broker uses the root w_0 and the customers signature to verify the received paywords. If the are validation is success, the broker deduct the spent amount from the customers account and credit to the vendor's account.

This protocol is offline system. Customer do not have to contact the broker for all the transactions other than beginning of the lifetime of each certificate [29].

Some of the drawbacks are,

- This system is also credit based system. Therefor customer account debit phase happens after spending some sort of time frame. So number of transactions can be done with insufficient funds.
- Paywords are customer and vendor specific and nothing can be done with other vendors with existing paywords [22].

2.5.4 NETPAY

Netpay protocol is mostly targeting the world wide web. The use of touchstone which is signed by the broker is required to verify the paywords by the vendor. To prevent the double spending, vendors are forced to pass the index which is signed by the vendors. Customer trust is not needed for this protocol [9].

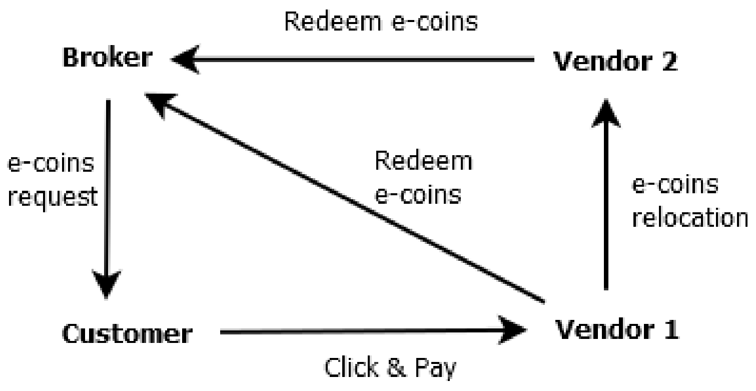


Figure 2.4: Netpay Payment Model

E-coins request: Customer has to send an integer n with a message before request service from the 1st vendor V1. The number of paywords in a payword chain is denoted by n. Same n is used as IDc to the broker also. The broker should have to completes 2 actions after that.

- Creates a payword worth debited money from the customer account.
- The touchstone T should be computed which consists of IDc and the W_0 for the chain.

Transaction: When the customer wants to purchase from vendor V1, customer sends a message. That message includes IDc, Payment
$P = (Wj, j), (Wj + 1, j + 1), .., (Wj + m - 1, j + m - 1)$
(m cents) and an order to the $V1$. $V1$ verifies the payment.

Paywords Relocation: If there is a customer who wants to make a purchase at $V2$, then the customer should sends IP address of $V1$, payment, order to $V2$ and IDc. To get the Index from $V1$, $V2$ transmits IDc and $IDv2$. Then $V1$ should have to signs the Index $= IDv1, IDv2, I$ where I is the last payword's Index. $V1$ transmit payword chain touchstone to $V2$. The Index can be use if any disputes among both vendors. $V1$ uses Index and $W0$ to verify the payment. If the payment verification is successful, it is stored and use with the broker as an offline transaction later. After this, the customer is supplied the purchased services or goods..

Offline Redeem processing: There is a process each end of the day between the vendor and the broker which transmit the touchstone IDc, IDv, and payment for each chain. The broker verifies the each received payword from vendor by generating hashes of them and counting the amount of paywords. After the successful validation, the vendor's account will be deposited with respective amount.

Netpay which is an offline protocol also protects the customer privacy. Customer forging and also double spending is also prevented on this protocol [9].

2.5.5 PPAY: Micro Payments for Peer-to-Peer Systems

Floating, self-managed currency concept is involved with PPay micropayment system. Therefore, broker involvement can be reduced [14] [8]. When peer act as a both buyer and vendor, the floating currency concept is more valuable float from one node to another without involving a broker [33].

The user who owned the coins managed all the security related features which refer as self-managed. In this way this protocol harnesses the available resources within peers.

A user U firstly purchases the digital coins from the broker B. U is the owner of the coins now.

$C = U, SnSKB$

Sn Uniquely identified serial number.

When U wants to buy from user V, U sends assigned coins to V. V is the owner of the coins now.

$AUV = V, seq1, CSKU$

seq1 represent the assigned sequence number which is always larger than previous one. Any time coin holder can cash the coins at the broker for flat fee.

Instead of cashing the coins, V can use to pay for another use X. Then V has to send reassignment request to U.

$RUVX = X, AUVSKV$

U then sends to V and X the new assignment with keeping a record of reassignment.

$AUX = X, seq2, CSKU$

Both V and X are responsible to validate the new sequence is greater than previous sequence. V can send a copy of reassignment to X if the x claims about not receiving it [33].

2.5.6 WHOPAY

WhoPay extends from the from PPay by extending architecture features. Therefor the coins lifecycle behaves same as in PPay. Customer should have to buy coins from broker before spending. The vendor can become a customer using received coins or deposit on broker and get cash. The coins are transferable and expirable. Therefor to retain the value of the coins, it is required to revalidate with broker time to time. [32].

1. U purchases coin from broker
2. U issues coin to V
3. V transfers coin to W through U
4. W deposits coin at broker

Purchase: U, who wants to purchase coins from the broker, generates random private key and public key. After the generation of key pair, U sends the public key and the signed identification using kept private key to the broker. Broker validate the signature of the U, broker adds the received public

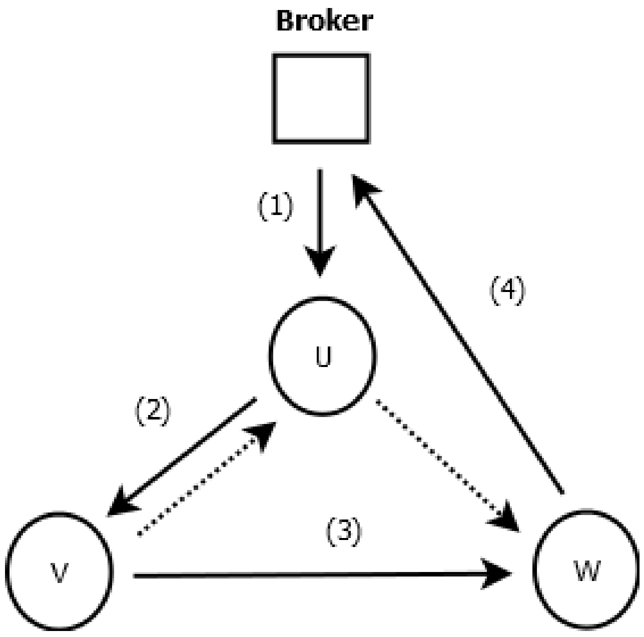


Figure 2.5: Whopay Payment Model

key to the list of valid coins. Those coins also signs using its private key. Then broker sends it back to U. U verifies the broker's signature and completes the transaction.

Issue: When U want to issue a coin to V, V should generates a new random private key and public key pair. Then V stores the generated private key and sends the newly generated public key and challenge to U. Then U answers the challenge and sends the broker-signed coin. This challenge is to prove the owner of the coin. Then U have to updates its coin binding list. That is a randomly selected number and an expiration date. U have to sign the binding with private key. Then U sends the signed binding to V. V verifies the received signature and complete the transaction.

Transfer: If the W wants to get paid from V, W should have to generate new public key and private key pair. W keeps newly generated private key himself and sends newly generated public key to V. V request a transfer request from U attaching Us public key and Ws public key. This transfer request should have to sign with Vs private key (to prove Vs holdership of the coin) and Vs group private key (to make sure the fairness of the system).

U verify the received transfer request and sends the broker-signed coin to W. U sends the answers to the challenge which received with transfer request from W to prove that the U is the owner of the coin. U has to update the coin binding list with updating to Ws public key, a suitable expiration date and an incremented sequence number. U has to sign the binding and sends to W to prove the transfer of the coin. W verifies the signature and complete the transaction.

Deposit: W should send a deposit request to broker by identifying the coin relation. The deposit request should be signed with Ws private key (to ensure Ws holdership) and Ws group private key (to help ensure the fairness of the system). Broker validate the signatures of the deposit request and transfer the money to W if the verification success. [32].

2.5.7 PEPPERCOIN

Efficient processing is the key for the micro payment scheme. Efficient way to process them is the Aggregation of many small valued transactions to few larger value transactions. No aggregation meant every transaction should go through whole cycle. Therefore, the cost is high.

Session level/merchant level aggregation which uses one macro payment on the customer after some sort of period. But then there should be more purchases from same client to same merchant.

Aggregation by Intermediation Intermediary which all the consumers and merchants will interact, will handle all the consumers and merchants debit/-credit as a one macro payment by tracking all the history of individual.

Papercoin method uses Universal aggregation/cryptographic selection. Each merchant processes the small valued transactions directly using cryptographic software. First it checks the validity of transaction by checking consumers transaction. If it is validated the goods/services will be delivers to the consumer.

Then the software checks whether the transaction to process a micropayment or macro payment. If it qualifies as a micropayment no further process other than logs it. If it qualifies as a macro payment the merchant will deposit that payment with his bank.

In here merchant expect to receive same amount on the average. The fraction of micropayments will be considered as a one macro payment and it depend on the size of micropayments.

Merchant will pay only single transaction fees for several micropayment transactions.

The qualification process is cryptographic and other parties can not affect the decision. Each micropayment evaluates for selecting and therefor merchant do not need to have a track of history.

Peppercoin system ensure that consumer is billed only what he paid. Deposits made by the merchant are always macropayment and consumers are billed for the amount they have spent at all the merchants along the considerable period.

Pappercoin can be implemented with digital signature or without it trusting consumer sign [28].

2.5.8 Swing-Pay

Swing-pay is a digital card module which uses NFC and biometric authentication for Peer-to-Peer Payment. It is an online micro payment scheme and it's main approach is to provide a contactless payment option with biometric authentication. This scheme presents a module which one card can communicate with the other using Near Field Communication technology to transfer money from payers bank to payees bank by digital means. The main objective is to eliminate the physical cash and to support all most all the payment types [12].

This scheme uses a module which is supportive for Near field communication, Finger print scanning and GSM connection. When a user buys a module first time, he has to register it with cloud server providing bank account details. One time password will send to registering mobile number and depends on the validation results makes the success or failure of the process.

When a payer want to make a payment to payee, payee send the payee's unique identification number to payer through NFC. Then payer sends the payee's unique id and payer's unique id with amount to pay to the cloud server through SMS service. Cloud server verifies the information received and sends to payer's bank. Bank will do the payment transaction and then payee's bank will notify the payee about receive of the payment [12].

2.6 EMV Tokenised Payment to offline environment

Existing EMV architecture involve with online connectivity during a transaction. But this has may drawbacks as most of the places does not have proper online connectivity. Therefore, there is an offline contactless mobile payment protocol based on EMV tokenisation. [15].

In the offline transaction environment considered in this proposed scheme [15], online connectivity to reach the authorising entity is not possible on either the secure element or the terminal. There are only two parties involved during transaction which are the secure element and the terminal. Therefore, in such a scenario the terminal needs to decide whether to accept or decline a tokenised transaction.

As the payment transaction is happened on offline environment, it is necessary to securely conduct the payment and the communication between the secure element and the terminal. The payment settlement phase is carried out when the terminal has online connectivity at a later time [15].

2.7 Summary of Micro Payment Schemes

This section provides comparison using advantages and disadvantages of existing micro payment protocols.

Table 2.1: Comparison of existing payment gateways.

Payment Gateway	Advantages	Disadvantages
Millicent	Allow repeated buying option for same vendor. Double spending prevent without contacting broker online. So less overhead. Allows payment to be validated in less overhead.	Seller specific repeated buying option only. Broker must be online when the customer goes to buy something from new vendor. Transactions are very complex when the customer and the broker has different brokers.
Mpay	Has offline capability. Uses one or no public key operations and so the transaction cost is low. Easy to use for customers.	Have to obtain a certificate daily. Each transaction should verify transaction daily limit exceeded or not. Transaction depends on Issuer availability.
Payword	Offline system. Customer only needs to contact the broker at the beginning of certificate lifetime.	Credit based scheme - Provide more opportunity to purchase goods with insufficient funds. Paywords are vendor and customer specific.
Netpay	Protects customer privacy. Prevents double spending. Can perform many purchases from many vendors. No customer trust required.	Credit based scheme

Peppercoin	Efficient processing low cost Variable sized payments Scalability Non-interactivity.	Better for the vendors having averagely same small valued transactions.
PPay	Allow the use of layered, transferable coins. Best for peer to peer networks. Broker performance is crucial due to self-managed coins.	Users who owned transferred coins may have to cash or create new coins if the originating user is not on-line. Peers will need to managed the coins. Double spending detection only.
WhoPay	Provide full anonymity to holder of coin's owner. Better implementation than PPay for P2P network which required anonymity. Coins are transferable.	When the user is offline the load of the broker is more high. Double spending prevention is more complex. Double spending detection only.

2.8 Outstanding Challenges

2.8.1 Security Challenges

Generally, Security is a preliminary concern for online payment systems. The report by CyberSource indicates that the fraud rate of online shopping with credit card is around ten times higher than physical card fraud rate [1]. Several studies by FICO indicates that US credit card fraud rate is increasing rapidly [6].

This trend become more worst with bitcoin also. Because of the unavailable of dispute resolution mechanism and the fraud protection, the transaction fees are very low. There are several incidents of success hacking attempts and stolen of Bitcoins from Bitcoin exchangers, market places and wallet services.

These results lost of hundreds of thousands of customers Bitcoins. One study document indicates that recently established 18 of 40 bitcoin exchanges closed already [21].

There are malware programs designed to steal bitcoins from users devices. Some of them are keyloggers and most of them successfully bypass most of the anti-viruses also [6].

Micropayment are normally vulnerable because of the security is not an isolated feature. Adding fraud mechanism with institution can reduce the brokers profit margin and increase the transaction fee. This is justifiable with macropayments but not with micropayments.

Because of these concerns, adding fraud mechanism can be impact to system usability and cognitive burden on the user. For very low value transaction users may not like to go for an extra hassle of mechanisms like two factor authentications.

2.8.2 Legal and Ethical Challenges

Legislation protection for micropayment systems is also a required factor like other real-world applications. Because doing businesses in unethical manner like delivering defected goods, cannot be covered with technology alone. There is a legislator role for balancing interest of users and merchants. Merchants and Brokers should decide how to accept risk and absorb more customers with confident.

Legislators not only have to make regulations in payment systems but prevent customer dissatisfaction also.

There are several scenarios identified which gather large number of user data and monetize them [6]. There are only few micropayment systems which offer anonymity from both merchant and broker. Users purchase history like individual articles, audio recordings and videos can expose users political aspect, sexual preferences or religious interesting.

It is a legislative and technological challenge when discovering a scalable micropayment scheme which protect payers privacy but preventing anti-money laundering activities and terrorism financing.

2.8.3 Psychological Challenges

One of research paper [6] recommends for systems developers, first rather than focus on technological innovation alone, it is imperative to recognize mental transaction costs. Technology may then be applied to alleviate these costs.

Some donation campaigns are using micropayment systems with labeling amount which payers can donate them. Those labels indicate real world scenarios and amount paid is identical to label indicates rather than pointing exact value. Example of labels with amount are mentioned in table 2.2 [6].

Table 2.2: Example of labels with amount.

Label	Amount	Currency
Beer	3.50	USD
Buck	1.00	USD
Cent	3.50	USD
Cerveza	1.50	USD
Coffee	1.50	USD

One of the major impact when considering phycological aspect on use of micropayment systems is branding and quality control. When a brand delivers good quality goods for money consistently, payers may be more inclined to trust and interest its product and make less mental effort in choosing to buy it. One way to implement this would be the easy refund option on purchased and reward quality goods [6].

Researches related to mental effort or phycological models for micropayment are less. Understanding the psychology behind micropayments will not only improve system usability but also assist in crafting appropriate business models and successful deployment strategies [6].

2.8.4 Deployment Challenges

One of research paper [6] mention that "there are important open research questions regarding pricing for a micropayments ecosystem. One of major Question is the pricing threshold at which micropayments become viable. While there is currently no definite answer to this question, there may be some evidence for it. The Chicago Sun-Times launched a Bitcoin-based micropayment donation option for 24 hours in February 2014 and collected over 700 payments, ranging from a penny to over $ 1000. 63% of these payments were for 25 cents".

Novel business models are proposing by several companies to get digital goods monetizing than use of traditional subscription payment model.

For a successful mass deployment of micropayment system, in perspective of deployment challenges, there are certain properties to consider. For example, a micropayment solution should be able to interconnect with existing payment solution infrastructures and should be inter-operable across a wide range of merchants. Standardization of payment protocol is getting much required task for this scenario.

Mobile phones are an ideal candidate in real world environment as smartphone usage is high, and all most all smartphones now support considerable more resources in computing, memory and bandwidth. Gartner study [4] about e-commerce revenue in the United States, predicts that by 2017, mobile payments will make up to 50%. Goldman Sachs predicts that by 2018 mobile phone purchases will constitute 50% of ecommerce globally [3].

Customers or users more likely to adopt to payment gateways if they are more trust worthy and have a positive reputation [6]. Merchants are waiting for a payment gateway with considerable number of users. Therefore, integrating payment gateway with mobile devices or with social networks will make it more familiar with existing market.

Chapter 3

Protocol for e-Cash using Standard Digital Signatures

3.1 Notations

The protocols developed in this work shall use following notation scheme:

- Digital signing of a message m by a signer *Alice* using her secret key generating the signature σ: $\sigma = sign_A(m)$
- Verification of a digital signature σ on a message m by a signer *Alice* using her public key pk_A: $verify(pk_A, m, \sigma)$
- Generation of a pseudo random number n: $n = prng()$

The standard digital signature scheme $sign_{secret-_key}(message)$ used in the protocols described in this work is based on the RSA public key cryptosystem. The key generation for RSA is done as follows:

1. Select two large prime number p and q randomly and compute the prime composite n as $n = p \cdot q$ keeping the values p and q secret.
2. Compute the Euler totient function $\phi(n)$ as $\phi(n) = (p-1)(q-1)$ keeping the value $\phi(n)$ secret.
3. Select an integer value e such that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$.
4. Compute an integer value d such that $d \cdot e \equiv 1 \; mod \; \phi(n)$.
5. the public key for signature verification is (e, n) while the secret key for signing is d.

3.2 Registration

First phase of the proposed payment scheme is registration process. The steps are as below.

1. $C \rightarrow B$: client request to bank for registration, Bank account number

2. $C \leftarrow B$: $n = prng()$
   n is a nonce

3. C : $generate(pk_C, sk_C)$
   Customer generate his own public key and private key pair.

4. $C \rightarrow B$: $\sigma_0 = sign_C(n), pk_C$

5. B : $verify(pk_C, n, \sigma_0)$
   this verification assures the authenticity of client and the freshness of the transaction

6. $C \leftarrow B$: $\sigma_1 = sign_B(pk_C)$

7. C : $verify(pk_B, pk_C, \sigma_1)$

3.3 Spending e-cash

3.3.1 Obtaining e-cash from a bank

Second phase of the proposed payment scheme is obtaining e-cash process. The steps are as below.

1. $C \rightarrow B$: client request to bank for e-cash

2. $C \leftarrow B$: $[m, \sigma_0 = sign_B(m)]$
   m is the e-cash with a unique serial number, value, and other details as decided by the bank

3. C : $verify(pk_B, m, \sigma_0)$
   this verification assures the authenticity and integrity of the e-cash received from bank

3.3.2 Spending e-cash at a merchant

Most important phase of the proposed payment scheme is spending e-cash at merchant. The steps are as below.

1. $C \rightarrow M$: client request to merchant for a transaction of value m in e-cash
2. $C \leftarrow M$: $M_{I_D} = MerchantID$, $n = prng()$
   n is a nonce
3. $C \rightarrow M$: $\sigma_1 = sign_C(n), [m, \sigma_0 = sign_B(m)], [M_{I_D}, Timestamp, \sigma_2 = sign_C(m + M_{I_D} + Timestamp), [pk_C, \sigma_3 = sign_B(pk_C)]$
4. M : $verify(pk_B, pk_C, \sigma_3)$
   this verification assures the validity of the public key of client
5. M : $verify(pk_C, n, \sigma_1)$
   this verification assures the authenticity of client and the freshness of the transaction
6. M : $verify(pk_B, m, \sigma_0)$
   this verification assures the authenticity of client and integrity of the e-cash
7. M : $verify(pk_C, m, \sigma_2)$
   this verification assures the validity of the non-repudiable claim made by client of spending the e-cash m

Merchant M is required to maintain a registry of e-cash receipts as

e-cash details	non-repudiable claim by client (spender)
m	σ_2, $Timestamp$
...	...

Details in the table are used to assist in the tracking of clients who double spend e-cash. Storing the values and signed values of the merchant id and time stamp provides the merchant inability to deposit same e-cash twice at bank and two different merchants to share e-cash and deposit separately claiming customer spend on both of them. Innocent clients have evidence to be protected with that.

3.3.3 Depositing e-cash at the bank

After collecting e-cash from customers, merchant deposit them on the bank. The steps for deposit e-cash at the bank are as below.

1. $M \rightarrow B : [m, \sigma_0 = sign_B(m)]$
2. $B : verify(pk_B, m, \sigma_0)$
   this verification assures the validity (authenticity and integrity) of the e-cash

Bank B is required to maintain a registry of spent e-cash

e-cash details	identification of merchant (depositor)
m	M
$\dots$	$\dots$

Details in the table are used to detect any double spent e-cash and then in the tracking of clients who acted illegally.

3.4 Detailed protocol

3.4.1 Registration with the bank

Customer has to install the mobile application for smart phone and complete the initial registration process within the application. To complete the initial registration, customer has the capability to generate his own public key and private key pair. The customer sent the request to bank informing about registration request message. Customer also sends the bank account number which is going to be linked with this scheme.

The Bank verify the received bank account number and replies with random one time password to customer mobile number which is associated with customer provided registered bank account, as a text message via short message service. Then customer should sign received nonce value using his generated private key and sends back to bank with his generated public key. Bank verifies the signed value and then replies the signed public key of the customer using bank private key.

3.4.2 Obtaining e-cash from a bank

Customer pay for e-cash which the customer request from bank using Internet banking, ATM or any other KIOSK devices. After the completion of payment, Bank replies to the customer with respective worth e-cash and signed value of e-cash using bank private key. This m contains with a unique serial number, value of each point and other details decided by the bank.

Customer can verify the received e-cash m using bank public key. verified e-cash stored in customer smart phone.

3.4.3 Spending e-cash at a merchant

When customer wants to pay a merchant, first customer request transaction value from merchant. Merchant sends the transaction value with the random nonce value. Customer sends the signed random value received from the merchant, e-cash which worth transaction value, signed value of the e-cash by customer, signed value of the e-cash by the bank, public key of the customer and signed value of the customer public key by bank's private key,

This random value is to verify the freshness of the transaction and to make sure that the customer is having the respective private key of the shared public key. Otherwise customer can send different customer public key trying to pretend as different person.

Merchant first verifies the received customer public key and then verify the received random value and it's signed value also. After successful completion of verification, merchant verifies the e-cash using both bank's signed value and the customer's signed value. This verification covers the authenticity of the customer, non-repudiable claim made by customer and the integrity of the e-cash.

Merchant required to maintain a history and a registry of received e-cash with non-repudiable claim made by customer which is the signed value of the e-cash using customer's private key to assist in the tracking of double spending e-cash.

3.4.4 Depositing e-cash at the bank

Registered merchant can request from bank to initiate a deposit of e-cash. Then merchant sends the stored e-cash and the signed value of e-cash using bank's private key. Bank verify the signed values and then assure the authenticity and the integrity of the received e-cash. Bank stores the e-cash and the merchant identification details.

Storing of the deposit details ensure that merchant cannot deposit same e-cash twice and verification of the e-cash using signed values assure the validity of deposited e-cash.

3.5 Analysing the properties

3.5.1 Double spending

If a customer tries to use same e-cash twice to pay for several merchants it detects when the second merchant tries to deposit it at the bank. It is not just a detections, but there are strong evidence about who paid this e-cash. There is a merchant maintain registry which contains the non-repudiable claim made by customer when making the payment by signing the e-cash with customer's private key.

There is a possibility to a different merchant to deposit same e-cash again saying customer has spend same e-cash on him also by getting shared the valid e-cash and signed value of it from real merchant who get paid from customer.

There is also a possibility to a merchant to deposit same e-cash again saying customer has spend on him again same e-cash.

We can mitigate this by storing customer signed random nonce value which provides by merchant at spending e-cash on merchant. Therefor a legal authority has a strong evidence to prove the validity of merchant's statement. Merchant already stores the non-repudiable claim made by him, so customer cannot decline the transactions he has made once but twice.

The updated version of the merchant registry is as below.

e-cash details	non-repudiable claim by client (spender)	nonce value
m	σ_2	n, σ_1
...	...	...

3.5.2 Non-Repudiation

Customer signs the e-cash using customer private key before sends to the merchant. Merchant can verify the customer signed e-cash using customer public key which has already verified using bank's public key.

Since the customer is the only one with access to the private key used to apply the signature, customer cannot later claim that it was not him/her who applied the signature.

3.5.3 Authentication

Customer provides the customer public key which is signed using bank private key. Merchant can verify the signature and authenticate the customer. Customer provides the signed random number which is provided by merchant also.

Therefor merchant can verify that customer actually holds the private key which is related to the provided bank signed customer public key.

since the customers unique private key was used to apply the signature, merchant can be confident that the customer was the one to actually apply the signature.

3.5.4 Integrity

Customer provides the e-cash with signed value of e-cash using bank's private key. The merchant can assure the integrity by verifying the signed e-cash using stored public key of the bank.

3.5.5 Availability

Our proposed scheme is not using any third-party service during transaction phase. Therefor it does not depend on the availability of the third-party services like most of the other payment gateways.

Our proposed scheme works in off-line way. Therefor the availability of the system is almost 100 percent assuming the mobile devices of customer and merchant works in good condition.

3.5.6 Transferability

Our proposed scheme is not using digital coins which are customer specific or merchant specific. Therefore, the transferability is applied to any party. Customer can transfer his coins not only to a merchant but also to another customer who can act as merchant at that time. Same applies to the merchant as the merchant can become a customer anytime while transferring his coins to another merchant.

3.5.7 Payment mode

Our proposed scheme is working on off-line mode as it does not use any on-line third-party involvement. Therefor it does not require any on-line communication also. Every validation, authentication and calculations are occurred within the device and communication is required between two smart phone devices only.

3.5.8 Usability

The smart phones are becoming popular and user-friendly every day. Our proposed scheme is using a smart mobile device which is very much familiar with most of the people now a day. Although the first time when registration, customer will not have to visit the bank.

When customer making the payment, none of the third-party services required. Therefore, customer able to use this scheme whenever he/she wants to make a payment. Customer is having the capability to decide the amount of the payment which provide the ability to makes a partial payment or over payment if decided to give a tip.

Other than selecting the amount to pay, mobile application will do all the calculations and validation itself which make customer to not to involve any extra works.

3.5.9 Validation

The customer can validate the merchant and the merchant can validate the customer themselves real time before accepting or making a payment. Depending on the usage of the system it may or may not need to verify the merchant.

Verifier can do the verification real time but in off-line mode using received signed customer public key and stored banks public key. Received of e-cash is also signed using customer private key and only the verified customer public key can verify the e-cash signed message. Therefor the validation occurs before accepting the e-cash from the customer.

If the customer also required to verify the merchant, it is possible to send some nonce value and get the signed value of the nonce value using merchants private key and merchant's public key and it's signed value using banks private key.

3.5.10 Transaction cost

In the transaction phase, our proposed scheme communication, calculation and validation happens within customer and merchant smart mobile devices only. Therefore, the transaction fee does not depend on the any other services or third-parties.

Other than direct transaction fee, customer or merchant does not have to face any communication fee also. As our proposed scheme is work in off-line mode it does not use on-line/ Internet communication which makes the communication cost almost zero. None of the participant should have to pay indirect communication data cost to mobile communication service providers also.

Although the payment gateways do not cost for the incomplete transactions, there are indirect communication cost which can be considerable while participants are dealing with micro payments. But our proposed scheme almost does not belong to any of the direct or indirect transaction or communication cost while making a complete or incomplete transaction as it does not involve with third-party services and it works off-line mode.

3.5.11 Anonymity

Customer has to provides the identification details at the verification process before initiate the transaction. After the transaction completion merchant stores the non-repudiable claim made by the customer. The customer public key with customer information is already with the merchant also. Therefor the customer anonymity is not available with this standard signature based proposed solution.

3.6 Problems with proposed solution

3.6.1 Minor limitations

There are few minor limitation on the above proposed solution. Those are mentioned below.

1. Merchants have to maintain a growing registry of transaction details. However, this could be justified as a legitimate business requirement similar to issuing a receipt for a transaction and then maintain a receipt book with carbon copies.

   However, it should be noted that transaction receipts contain details of purchased items, person who made purchases and mode of payment but not of the identification details of payment if it was made in cash. Only when a purchase is made using credit/debit cards or cheques, such details are recorded.

   Therefore, this limitation negatively impact on the requirement for transaction anonymity.

2. Bank has to maintain a growing registry of spent e-cash details. However, this could be justified as a legitimate bank activity where all transactions with a bank are recorded for every bank client.

   As noted earlier, while the transaction records at bank contain details of transaction and value, they do not record the identification details of payment if it is made using cash. Therefore, this limitation negatively impact on the requirement for transaction anonymity.

3.6.2 Major limitations

Proposed solution has one major limitation and it is mentioned below.

1. Anonymity is not totally protected although partial anonymity is available.

3.7 Summary

This chapter proposed a solution for micro payment scheme using standard digital signature. This chapter also provides the analysis on properties of micro payment scheme and provides the pros and cons also. This proposed scheme is having minor and major limitations also. Major limitations are addressed in the next chapter with improved solution for the micro payment scheme.

Chapter 4

Protocol for e-Cash using Ring Signatures

The anonymity requirement will be satisfied by replacing the standard digital signature scheme $sign(\ldots)$ with a ring signature scheme $ringsign(\ldots)$, which is a form of group signature scheme without a group manager and not having a fixed group membership.

4.1 RSA based Ring Singature Scheme

4.1.1 Ring Signature Generation

A ring signature is generated by a signer who collects an l number of public keys of other signers. While it is not necessary to have the consent of the public key owners to use their keys, these public keys must be selected in a manner relevant to the application context. For example, when a user is engaged in an online transaction, it would be safe to use a set of public keys of users who are engaged in online transactions with the server at the same time.

1. Calculate a symmetric key k of appropriate length, which is a security parameter, using a cryptographically secure hash function $H(\ldots)$ as $k = H(m)$ where m is a construction based on an arbitrary collection of l number of public keys $Y_1, Y_2, \ldots, Y_l$. Each public key Y_i is a RSA public key of value pair (e_i, n_i). For example, this construction could be the concatanation of public keys as $Y_1 \| Y_2 \| \ldots Y_l$.

2. Select a random value r.

3. Select a random value x_i for every member of the ring except for the signer and compute the corresponding value y_i as $y_i = x_i^{e_i}\ mod\ n_i$. Let the corresponding value for signer be y_s, which is yet to be computed.

4. Let the ring equation be $C_{k,r}(y_1, y_2, \ldots, y_l) = E_k(y_l \oplus E_k(y_{l-1} \oplus E_k(\ldots \oplus E_k(y_1 \oplus r)\ldots))) = r$. Solve this equation for y_s. It should be noted that the signer's value y_s can be set any index location in the collection including 1. The equation is based on a symmetric encryption function E_k.

5. Compute x_s as $x_s = y_s^{d_s}\ mod\ n_s$.

6. The ring signature is the $(2l + 1)$-tuple $(Y_1, Y_2, \ldots, Y_l, x_1, x_2, \ldots, x_l, r)$.

4.1.2 Ring Signature Verification

Any user with the ring signature $(Y_1, Y_2, \ldots, Y_l, x_1, x_2, \ldots, x_l, r)$ will be able to verify the signature as follows:

1. Compute the set of values $y_1, y_2, \ldots, y_l$ as $y_i = x_i^{e_i}\ mod\ n_i$ where $Y_i \equiv (e_i, n_i)$.

2. Compute the symmetric key k as $k = H(m)$ where $m = Y_1 \| Y_2 \| \ldots Y_l$.

3. Compute the ring equation and verify that $C_{k,v}(y_1, y_2, \ldots, y_l) = r$ is true.

4.2 e-Cash Scheme using Ring Signatures

Registration process steps are same as proposed scheme using digital certificate. We have improved the obtaining e-cash from a bank process.

4.3 Spending e-cash

4.3.1 Obtaining e-cash from a bank

Steps for the obtaining e-cash from a bank is given below.

1. $C \rightarrow B$: client request to bank for e-cash

2. $C \leftarrow B : [m, \sigma_0 = sign_B(m)]$
   m is the e-cash with a unique serial number, value, and other details as decided by the bank

3. $C : verify(pk_B, m, \sigma_0)$
   this verification assures the authenticity and integrity of the e-cash received from bank

4.3.2 Spending e-cash at a merchant

Steps for the spending e-cash at a merchant is given below. This process uses ring signature instead of digital signature at some steps.

1. $C \rightarrow M :$ client request to merchant for a transaction of value m in e-cash

2. $C \leftarrow M : n = prng()$
   n is a nonce

3. $C \rightarrow M : \sigma_1 = ringsign(n), [m, \sigma_0 = sign_B(m)], \sigma_2 = ringsign(m), [pk_{C1}, \sigma_{C1} = sign_B(pk_{C1})], [pk_{C2}, \sigma_{C2} = sign_B(pk_{C2})], ..., [pk_{Ci}, \sigma_{Ci} = sign_B(pk_{Ci})]$

4. $M : verify(pk_B, pk_{Ci}, \sigma_{Ci})$
   this verification assures the validity of the public keys of group involve to ring signature

5. $M : ringverify(pk_{Ci}, n, \sigma_1)$
   this verification assures the authenticity of group of people involve to ring signature and the freshness of the transaction

6. $M : verify(pk_B, m, \sigma_0)$
   this verification assures the authenticity of client and integrity of the e-cash

7. $M : ringverify(pk_C, m, \sigma_2)$
   this verification assures the validity of the non-repudiable claim made by client of spending the e-cash m

Merchant M is required to maintain a registry of e-cash receipts as

e-cash details	non-repudiable claim by client (spender)
m	σ_2
$\ldots$	$\ldots$

Details in the table are used to assist in the tracking of clients who double spend e-cash.

4.3.3 Depositing e-cash at the bank

This phase is for depositing e-cash which are collected from customers.

1. $M \to B : [m, \sigma_0 = sign_B(m)]$
2. $B : verify(pk_B, m, \sigma_0)$
   this verification assures the validity (authenticity and integrity) of the e-cash

Bank B is required to maintain a registry of spent e-cash

e-cash details	identification of merchant (depositor)
m	M
$\ldots$	$\ldots$

Details in the table are used to detect any double spent e-cash and then in the tracking of clients who acted illegally.

4.4 Analysing the properties

Our solution based on ring signature also maintains the properties of the solution based on standard digital signature. Our first solution major issue was the anonymity and We have achieved the anonymity by using the ring signature.

4.4.1 Achieving anonymity using ring signature

Obtaining e-cash from a bank is not using ring signature as bank identification is not need to have anonymity. Therefor we use standard digital signature.

Spending e-cash process is the transaction phase we have to maintain anonymity and we use ring signature based signing.

When the sender uses the standard digital signature for signing and sending e-cash to receiver, receiver can verify and store the received e-cash signed information. This way the anonymity of the transaction is not available. But when the sender uses the ring signature to sends the e-cash by signing using ring signature, receiver cannot extract the identity information of the sender and expose later.

We use ring signature to sign the n; nonce value received from the merchant also. This way we protect the freshness of the transaction and the authenticity of the group of people involve to sign using ring signature.

Client provides the signed value of the each group members public key to ensure the validity of the public keys of the group. The same group of people should use to sign the e-cash so that merchant can assures the validity of the non-repudiable claim.

4.5 Problems of Ring signature based solution

4.5.1 Minor limitations

There are two minor limitations with our proposed scheme and those are mentioned below.

1. Merchants have to maintain a growing registry of transaction details. This limitation exist with this solution also. However, this have already justified as a legitimate business requirement similar to issuing a receipt for a transaction and then maintain a receipt book with carbon copies.

   Our standard signature base solution has to store the identification details of the payment also. But our solution based on ring signature does not reveal the identification details of the payment.

2. Bank has to maintain a growing registry of spent cash details. However, this could be justified as a legitimate bank activity where all transactions with a bank are recorded for every bank client. We can proposed the

bank to encrypt transaction details using bank public key while storing with bank database. Within the bank, transaction anonymity is negatively impacted.

4.6 Summary

Because of the loss of anonymity of the previous solution, this chapter proposed an improved solution for micro payment scheme using ring signature. This chapter also provides the analysis on anonymity with improvement.

Chapter 5

Analysing the protocol

This section models the proposed micro payment scheme client-side using CSP. The parts of the micro payment scheme protocol that cooperate with registration and redemption is excluded from the analysing model. With the analysing of the payment model, it is assumed that the correctness of the cryptographic hash function is guaranteed independently. This chapter is to prove that the analysing protocol is deadlock free and non-blocking protocol.

5.1 Communicating Sequential Processes (CSP)

This chapter uses communicating sequential processes to model the our proposed micro payment scheme protocol. CSP is a formal language and it is used for describing interrelations between different parties or systems using process algebra. Systems should have to modelled in CSP under two sections called processes and events. A process represents a entities of a modeling system. The communication between different entities which are processes categorised under events. In this section We have used syntax for modeling the processes and those are mentioned below.

In order to model the protocol following syntax are used. Suppose the model has two processes (P and Q), two events (a and b) and a channel c.

1. $a \rightarrow P$: a is an event which is performed by a process and then behaves as process P.
2. $P|||Q$: process P and Q are interleaving processes and these processes perform actions independently.

3. $c!x \to P$: a process sends a message with parameter x to another process by synchronizing on channel c and then behaves like process P.

4. $c?y \to P$: a process receives a message with parameter y sent by another process on synchronized channel c and then behaves like process P.

5. $a \to P[]b \to Q$: a process either performs event a and behaves as P or performs event b and behaves as Q.

In CSP, Channels are used to communicate messages between existing processes. In our model, we have used synchronous channels. With use of the Synchronous channels, the sender may not have ability sends a message if the receiver is not ready to accept the message. There are different tools available for checking and analysing the models used CSP as language. We are using Protocol Analysis Toolkit (PAT) to analyse the model.

PAT is a tool which supports composing, simulating and verifying different systems. It supports different modeling languages also. PAT allows to verify several type of properties for a modeling protocol and we are using to prove that the analysing protocol is deadlock free and non-blocking protocol.

5.2 Analysing Client Registration process with Bank

The Client first install the mobile application and then ask from bank for registration. Then Bank send nonce value to client. Client generates his own public and private key pair. Then signed the received nonce value and send it to Bank with generated public key. Bank verify the nonce and replied to client with signed client public key using bank private key.

1. Customer/ Merchant install mobile application which includes the Bank and the Legal Authority Public key

2. Customer/ Merchant generates public key and private key pair using his own mobile device and then inform the bank about registration through mobile application.

3. The bank will send random nonce (n).

4. Customer/ Merchant should have to sign it and then send it with generated public key to bank.

5. Bank verify the received signed nonce using received public key and then sign the public key of the Customer/ Merchant using bank private key. Bank Sends the signed public key to Customer/ Merchant.

6. Customer/ Merchant verify the received signed public key and then store it with mobile application.

CSP code for the registration process is given in appendix A.

PAT analyser is used to simulate and the generated simulation graph is shown in the figure 5.1.

PAT analyser verification function is used to simulate and verify the client registration process is non-blocking and deadlock free. It's result is shown in the figure 5.2.

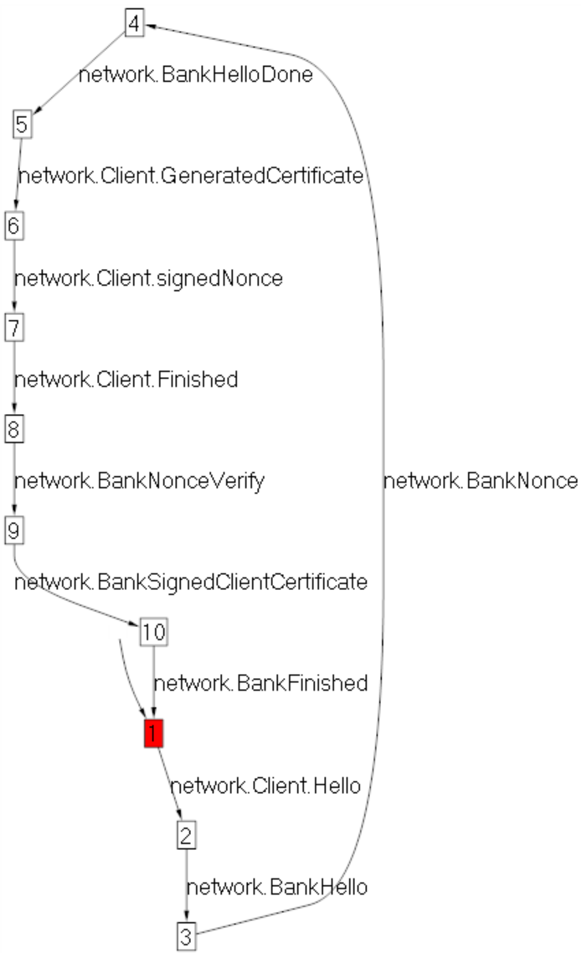
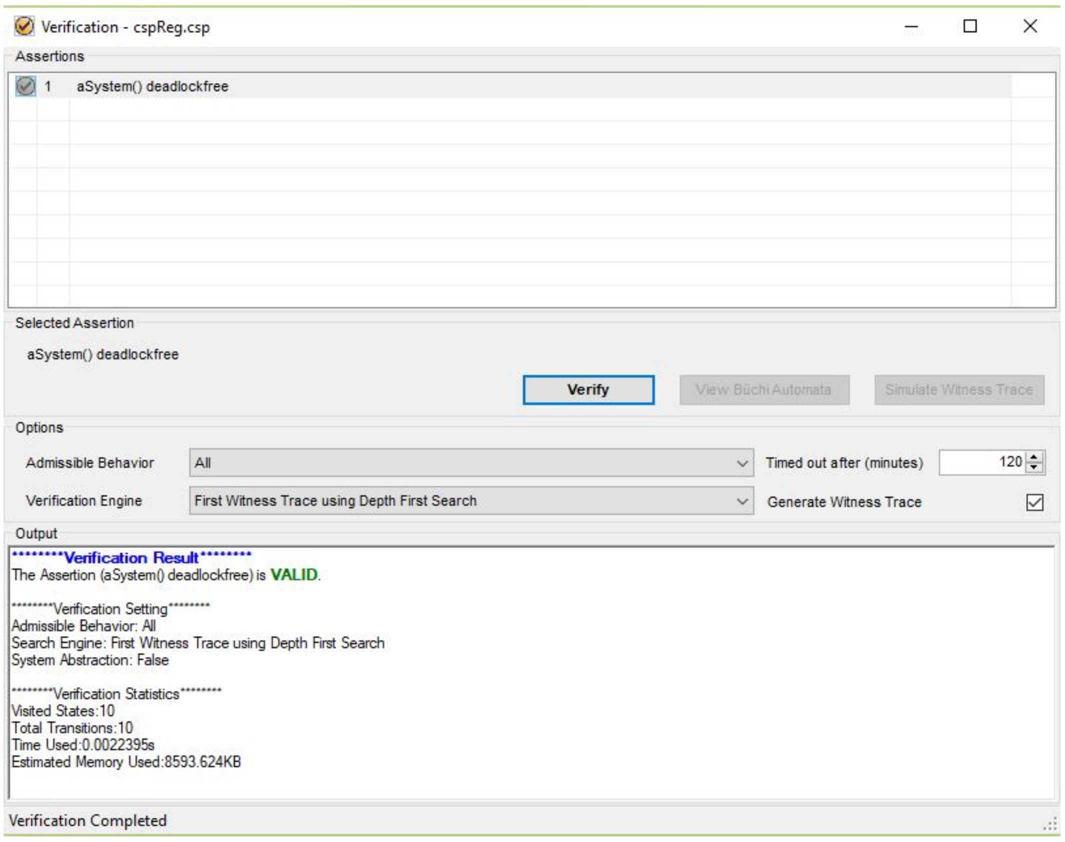


Figure 5.1: PAT Analyser Generated Graph for Registration Process



```
********Verification Result********
The Assertion (aSystem() deadlockfree) is VALID.

********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

********Verification Statistics********
Visited States:10
Total Transitions:10
Time Used:0.0022395s
Estimated Memory Used:8593.624KB
```

Figure 5.2: PAT Analyser Verification for Registration Process

5.3 Analysing customer e-cash obtaining process with Bank

The Customer first request for e-cash from the Bank. Bank replies with random nonce value. Customer sign the received nonce value and then send it to bank. Bank verify the received nonce value and then send the e-cash with signed value of it using bank private key.

1. Customer request e-cash from bank.
2. The bank will send random nonce (n).
3. Customer should have to sign it and then send it with customer public key to bank.
4. Bank verify the received signed nonce using received public key and then send the signed e-cash to Customer.
5. Customer/ Merchant verify the received signed e-cash and then store it with mobile application.

CSP code for the obtaining e-cash process is given in appendix A.

PAT analyser is used to simulate and the generated simulation graph is shown in the figure 5.3.

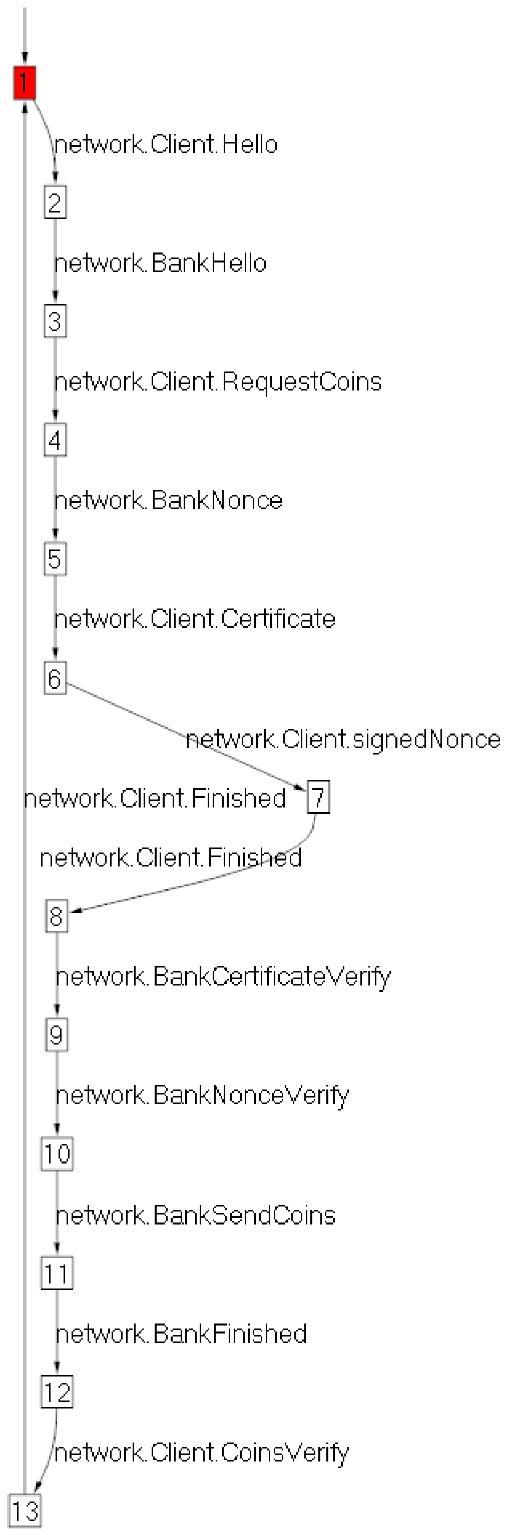
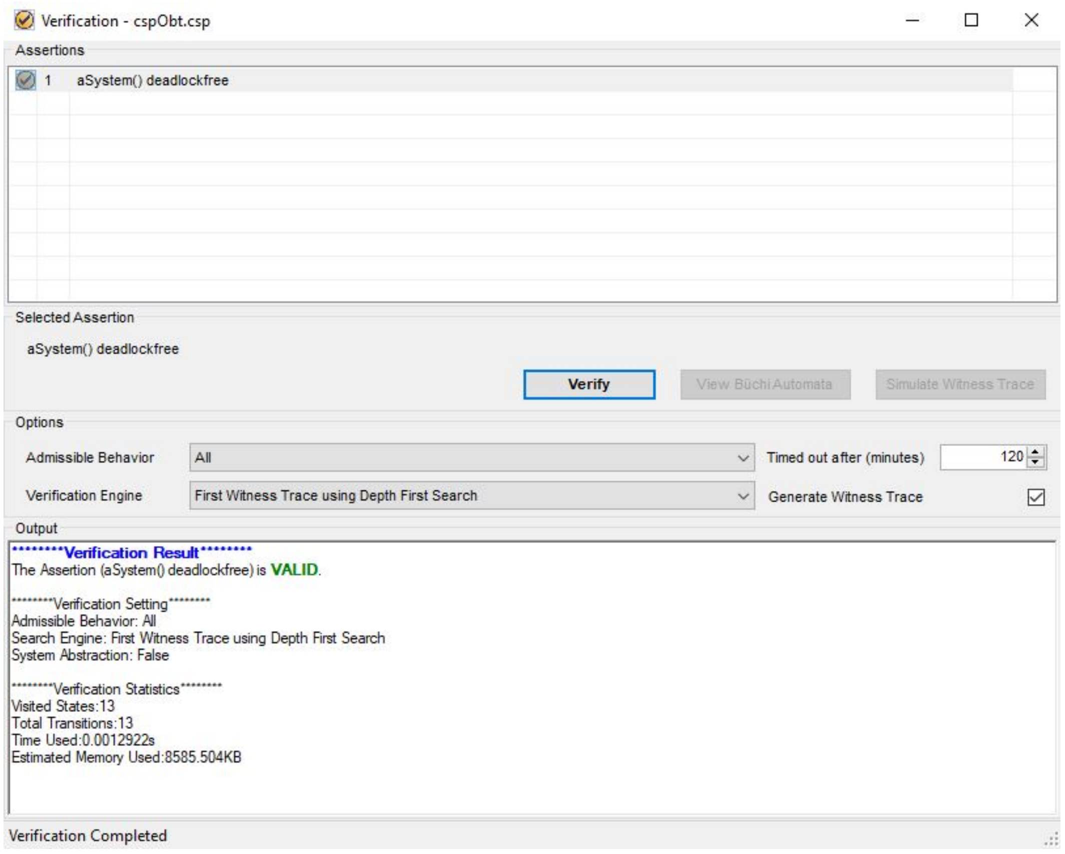


Figure 5.3: PAT Analyser Generated Graph for Obtaining e-cash Process

PAT analyser verification function is used to simulate and verify the e-cash obtaining process is non-blocking and deadlock free. It's result is shown in the figure 5.4.



Admissible Behavior: All
Timed out after (minutes): 120
Verification Engine: First Witness Trace using Depth First Search
Generate Witness Trace ✓

Output

```
********Verification Result********
The Assertion (aSystem() deadlockfree) is VALID.

********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

********Verification Statistics********
Visited States:13
Total Transitions:13
Time Used:0.0012922s
Estimated Memory Used:8585.504KB
```

Figure 5.4: PAT Analyser Verification for Registration Process

5.4 Spending e-cash at a merchant

The Customer first request transaction value from the merchant. Merchant replies transaction value with random nonce value. Customer sign the received nonce value and then send it to merchant with signed e-cash worth transaction value. Merchant verify the received nonce value, customer public key and the e-cash using bank public key and verified customer public key.

1. Customer request to merchant for a transaction of value m in e-cash
2. Merchant send random nonce (n).
3. Customer should have to sign nonce value and then send it to merchant with customer signed public key and signed e-cash.
4. Merchant verify the customer public key and then verify the nonce value and e-cash using verified public key of customer.
5. Merchant store the received e-cash within mobile application.

CSP code for the spending e-cash at merchant process is given in appendix A.

PAT analyser is used to simulate and the generated simulation graph is shown in the figure 5.5.

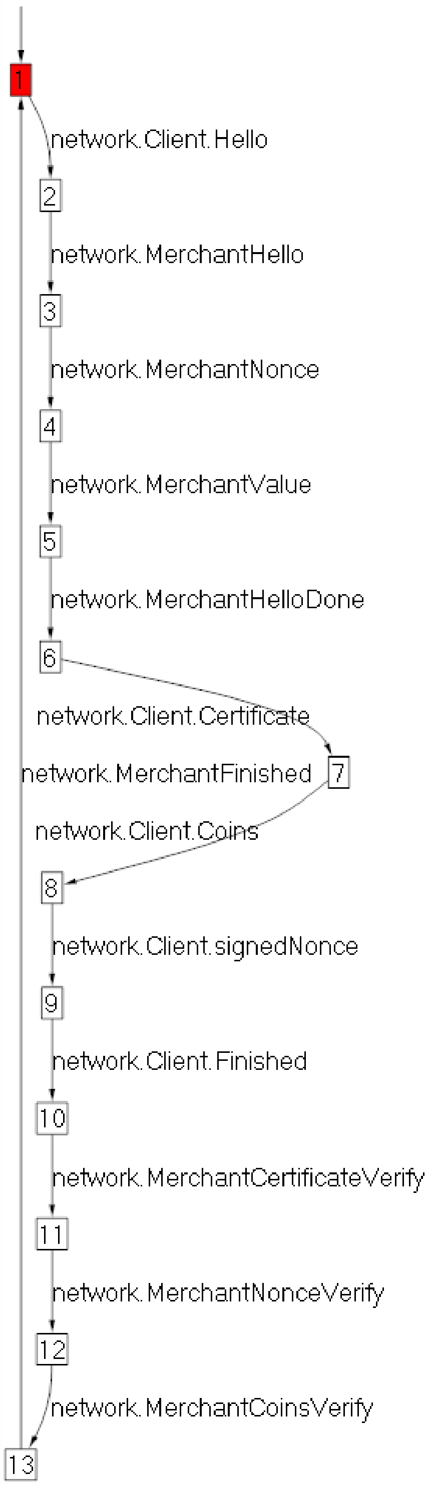
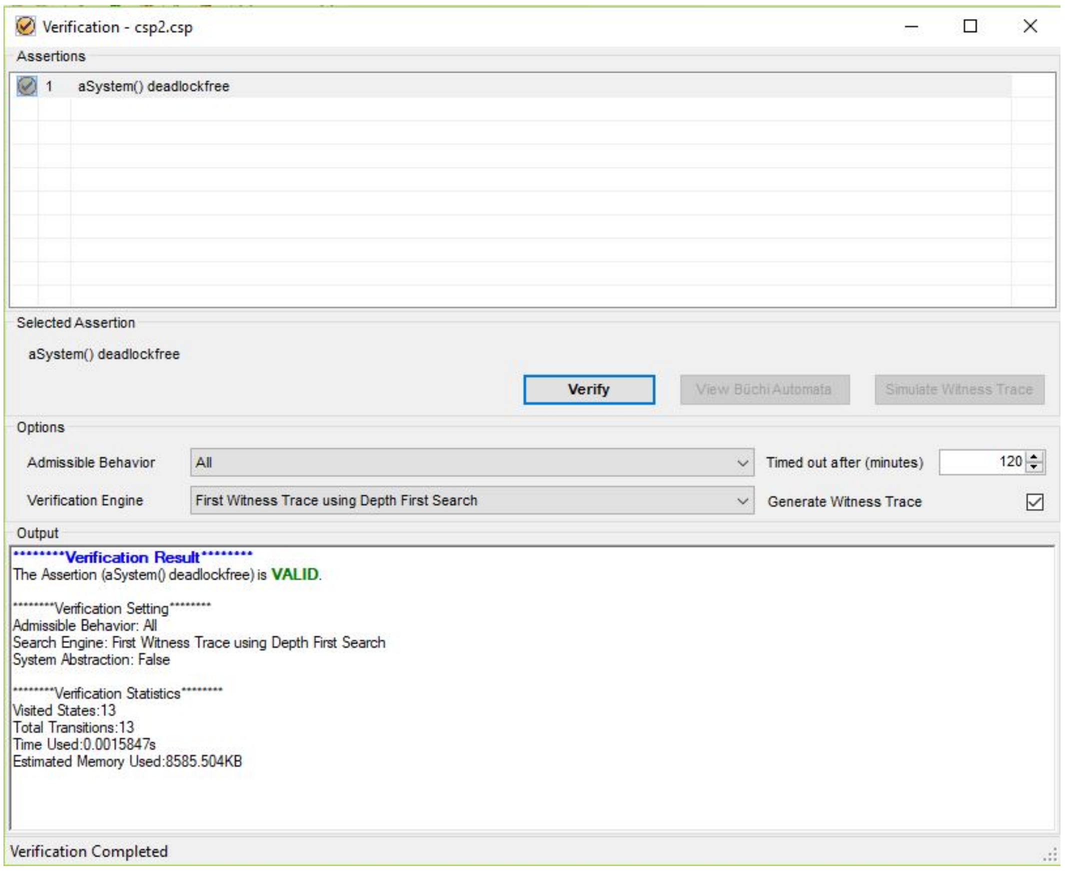


Figure 5.5: PAT Analyser Generated Graph for spending e-cash Process

PAT analyser verification function is used to simulate and verify the e-cash spending process is non-blocking and deadlock free. It's result is shown in the figure 5.6.



Admissible Behavior: All
Verification Engine: First Witness Trace using Depth First Search
Timed out after (minutes): 120
Generate Witness Trace ✓

Output

```
********Verification Result********
The Assertion (aSystem() deadlockfree) is VALID.

********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

********Verification Statistics********
Visited States:13
Total Transitions:13
Time Used:0.0015847s
Estimated Memory Used:8585.504KB
```

Figure 5.6: PAT Analyser Verification for spending e-cash Process

5.5 Depositing e-cash at the bank

The Customer first initiate the deposit request to the bank. Bank replies random nonce value. Customer sign the received nonce value. Then Customer send signed nonce, signed e-cash and customer public key to bank. Bank verify the received nonce value, customer public key and the e-cash using bank public key and verified customer public key. After successful verification, the bank deposits the amount worth received e-cash to customer account.

1. Customer initiate the e-cash deposit request to the bank.
2. Bank replies with random nonce (n).
3. Customer should have to sign nonce value and then send it to the bank with customer signed public key and signed e-cash.
4. The bank verify the customer public key and then verify the nonce value and e-cash using verified public key of customer.
5. The bank store the received e-cash at customer account.

CSP code for the depositing e-cash at bank process is given in appendix A.

PAT analyser is used to simulate and the generated simulation graph is shown in the figure 5.5.

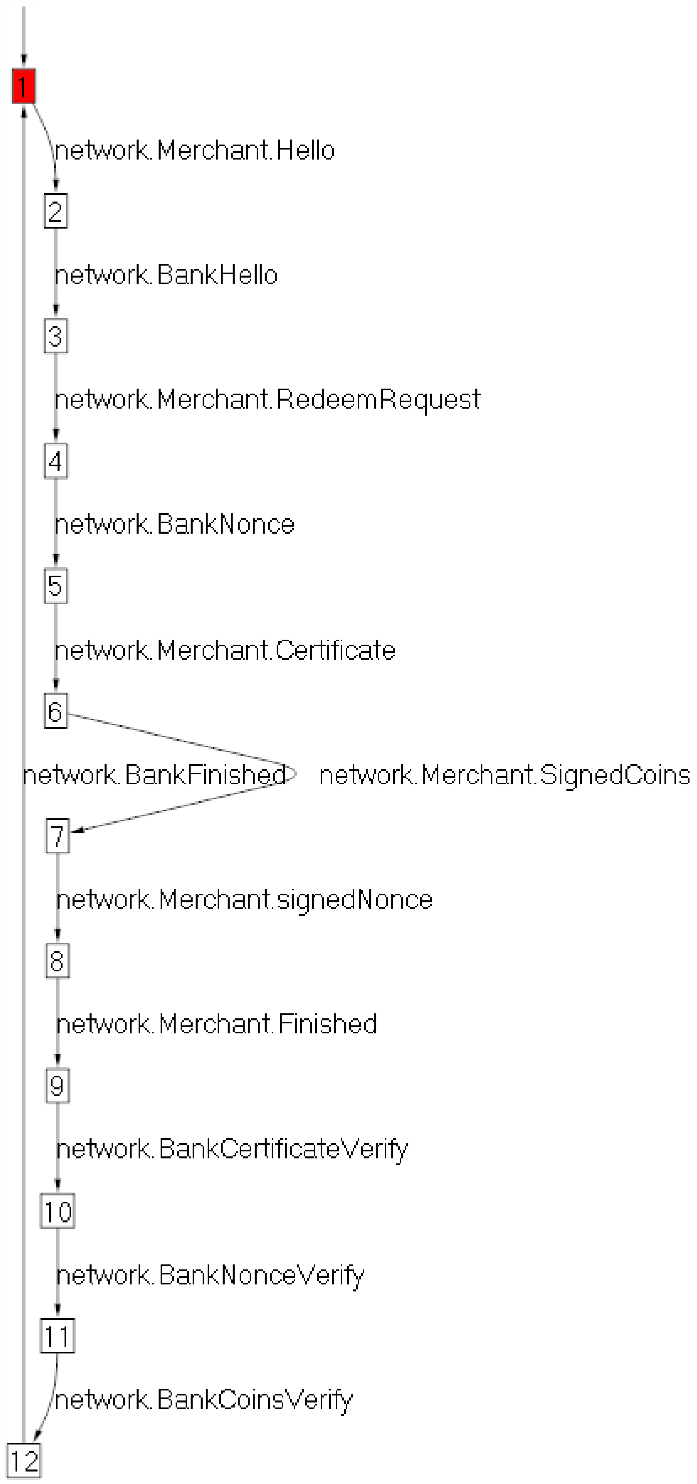
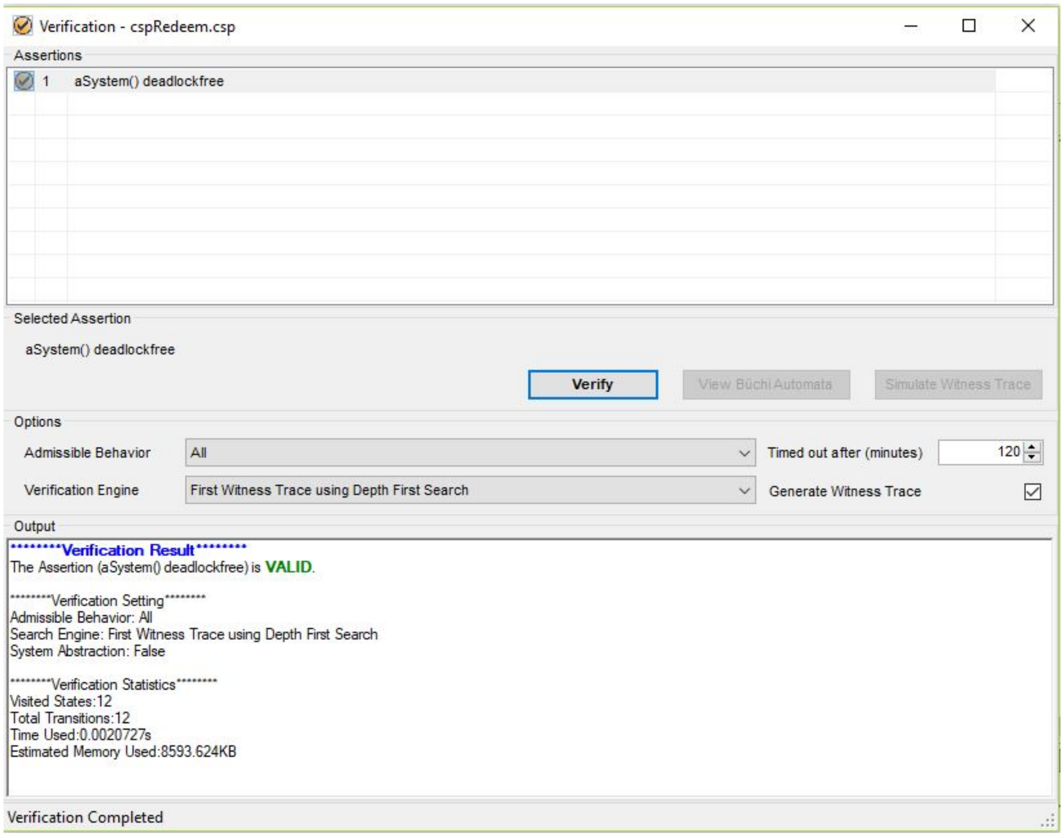

1
network.Merchant.Hello
2
network.BankHello
3
network.Merchant.RedeemRequest
4
network.BankNonce
5
network.Merchant.Certificate
6
network.BankFinished
network.Merchant.SignedCoins
7
network.Merchant.signedNonce
8
network.Merchant.Finished
9
network.BankCertificateVerify
10
network.BankNonceVerify
11
network.BankCoinsVerify
12

Figure 5.7: PAT Analyser Generated Graph for depositing e-cash Process

PAT analyser verification function is used to simulate and verify the e-cash depositing process is non-blocking and deadlock free. It's result is shown in the figure 5.6.



Admissible Behavior: All
Timed out after (minutes): 120
Verification Engine: First Witness Trace using Depth First Search
Generate Witness Trace: ✓

```
********Verification Result********
The Assertion (aSystem() deadlockfree) is VALID.

********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False

********Verification Statistics********
Visited States:12
Total Transitions:12
Time Used:0.0020727s
Estimated Memory Used:8593.624KB
```

Figure 5.8: PAT Analyser Verification for depositing e-cash Process

5.6 Summary

This chapter shows the process and verify that the proposed protocol is non-blocking and deadlock free protocol. CSP language and the PAT analyser used for model the protocol and verify the property assuming the correctness of the cryptographic hash functions are guaranteed independently.

Chapter 6

Proof-of-Concept Implementation

This section is about the Proof-of-Concept implementation of the solution that would run a client on a smart phone device for both a customer and a merchant with Bluetooth communication for exchanging messages during a transaction. We assume that the correctness of the cryptographic hash function is guaranteed independently. We will prove that the solution is possible to implement and usable practically.

Mobile application is developed using Android Java language and server side developments for bank environment is using PHP language.

6.1 Client Registration process with Bank

The Client first install the mobile application which already includes the Bank public key. Then customer should have to fill the registration form and submit to register with Bank. Then the registration process is handled by the mobile application itself. The process is as below.

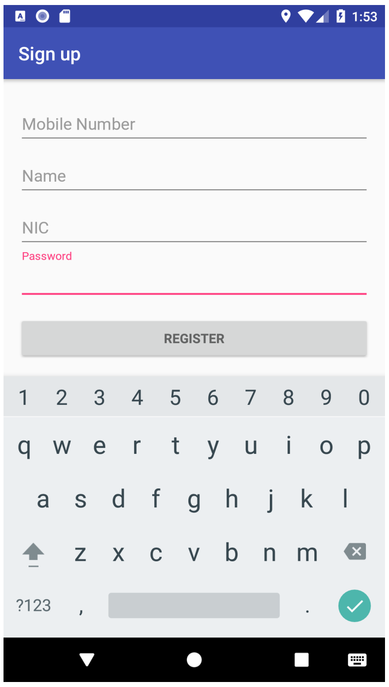
1. Customer/ Merchant install mobile application and fill the registration form.
2. Mobile application generates public key and private key pair using his own mobile device and then inform the bank about registration through mobile application.
3. The bank will send random nonce (n).

4. Customer/ Merchant mobile application sign it and then send it with generated public key to bank.

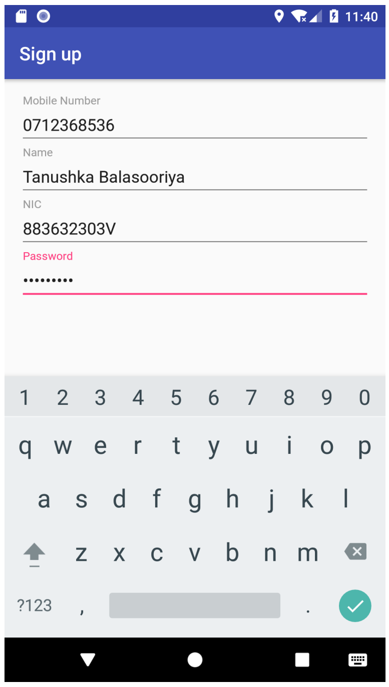
5. Bank verify the received signed nonce using received public key and then sign the public key of the Customer/ Merchant using bank private key. Bank Sends the signed public key to Customer/ Merchant.

6. Customer/ Merchant mobile application verify the received signed public key and then store it with mobile application for future use.

Mobile application interfaces for registration part is shown in the figure 6.1.



(a) Initial interface for registration



Mobile Number
0712368536
Name
Tanushka Balasooriya
NIC
883632303V
Password
•••••••••

(b) Filled interface for registration

Figure 6.1: Mobile application interfaces for Registration

After the successful completion of the registration with Bank, the client can act as a merchant or a customer. It depends on the situation which the client is selling a service/ product or purchasing a service/ product. If the client wants to act as a customer initially, it is required to top up the wallet.

Programming segments used to demostrate the processes are given in appendixes.

PHP code segment for signing a value using a private key and verifying signature using a public key are given in appendix B.

Bank registration process started with client request from bank to register. Then bank provides a random nonce value to client. Bank side PHP code segment for this process is given in appendix B.

Client mobile application receive the random nonce value and then sign it with generated keys and then reply back to bank with signed nonce value and customer public key.
Bank should verify the received signature of the nonce value and then reply back to customer with signed client public key. Server side PHP code for this process is given in appendix B.

Android code segment for mobile application to initialize the registration process and process the responses from bank queries is given in appendix B.

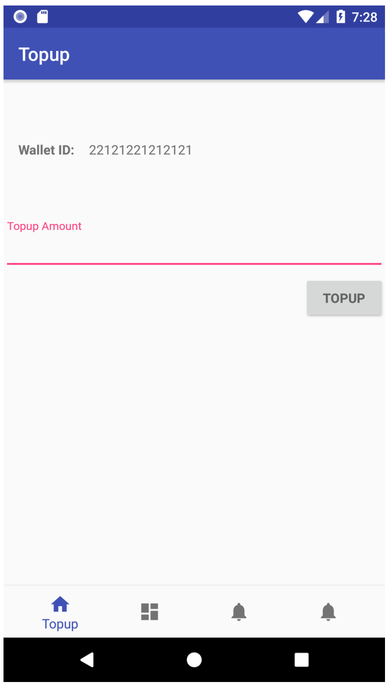
6.2 e-cash obtaining process with Bank

The bank may able to use several ways to top up the customer wallets. It can be using Internet banking bill payment option, linked bank account with wallet which have reloaded or any other way decided by the bank. In our PoC level, we assume that the bank has linking facility to link wallet to an existing bank account.

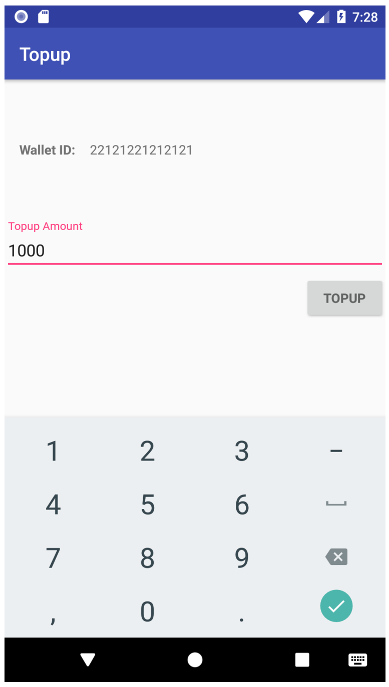
The Customer first request for e-cash amount from the Bank using mobile application. Bank replies with random nonce value to mobile application and then the received nonce value sign with customer private key and then send it to bank. Bank verify the received nonce value and then send the e-cash with signed value of e-cash using bank private key. Application stores the e-cash and signed value of it for future use.

1. Customer initiate the e-cash request using mobile application to bank.
2. The bank replies with random nonce value (n).
3. Customer mobile application receives the nonce value and then sign it using stored customer private key.
4. Bank verify the received signed nonce and then send the signed e-cash to Customer.
5. Customer mobile application verify the received signed e-cash and then stores both e-cash and signed value of it within itself for future use.

Mobile application interfaces for top up are shown in the figure 6.2.



(a) Initial interface for top up



(b) Filled interface for top up

Figure 6.2: Mobile application interfaces for top up

Bank top-up process code segment using PHP language is given in appendix B.

Client side top-up process code segment using Android language is given in appendix B.

6.3 Customer payment process with Merchant

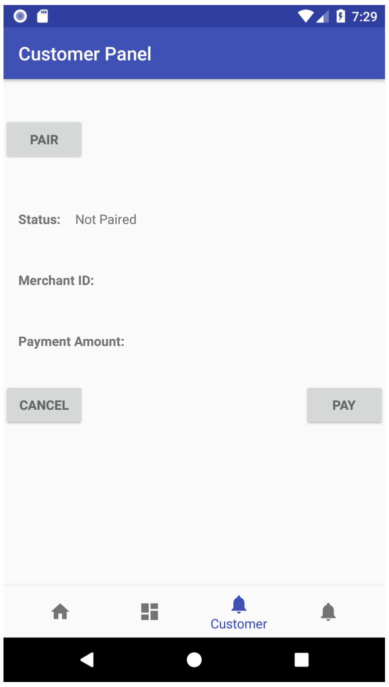
The Customer first pair the mobile device with merchant mobile device using application provided interface. Our PoC application uses bluetooth as a communication medium for offline transactions.

Customer and Merchant initial interfaces before starting the payment process are shown in the figure 6.3.

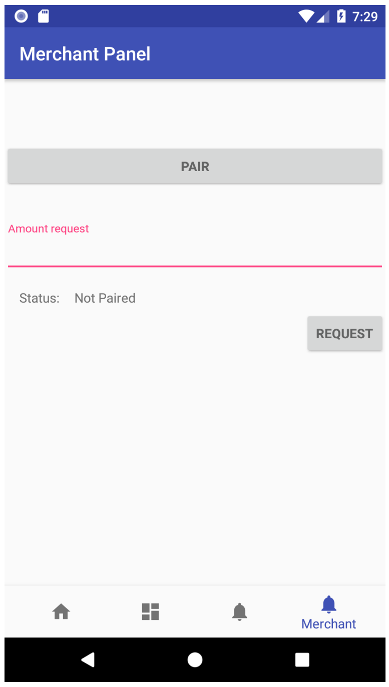
After a successful pairing of both devices, merchant sends the transaction value and mobile application append a random nonce value with transaction value. Customer application receive the value and displays it to the customer. After that the customer verify the amount. If customer accept the amount, customer mobile application sign the received nonce value and then send it to merchant with signed e-cash worth transaction value, customer public key and signed value of public key. Merchant verify the received nonce value, customer public key and the e-cash using bank public key and verified customer public key.

1. Customer initiate pairing of his mobile device with merchant mobile device.
2. Merchant accept and get paired using same communication medium.
3. Merchant sends random nonce (n) with transaction amount to get paid.
4. Customer accept the received amount and then application automatically sign the received nonce value.
5. Customer mobile application send signed nonce to merchant with customer signed public key and signed e-cash worth transaction amount.
6. Merchant application verify the customer public key and then verify the nonce value and e-cash using verified public key of customer.
7. Received e-cash get stored within mobile application for future use.

Customer and Merchant initial interfaces before start payment process are shown in the figure 6.4.



(a) Customer



(b) Merchant

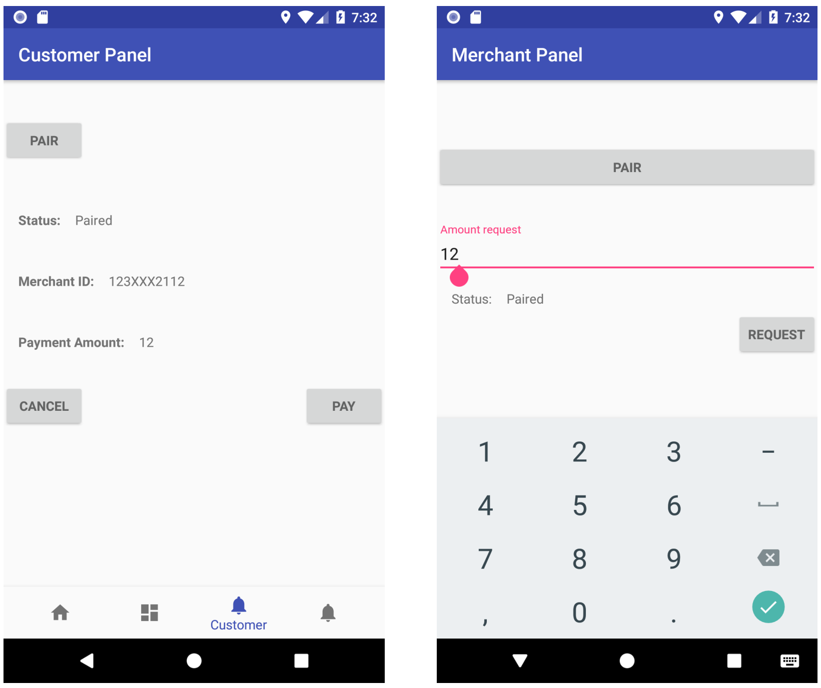
Figure 6.3: Customer and Merchant initial interfaces before starting payment process

Customer initiate the device pairing request and the payment request both. Code segment in appendix B shows the customer process during customer and merchant payment transaction using Android language.

Code segment in appendix B shows the merchant process during payment transfers between customer and merchant using Android language.

6.4 Withdraw e-cash to a bank account

The Customer first initiate the withdraw request to the bank with account number to deposit and the amount to withdraw. Bank replies random nonce value. Customer application sign the received nonce value send signed nonce, signed e-cash and customer public key with signed value of public key to the bank. Bank verify the received nonce value, customer public key and the e-cash using bank public key and verified customer public key. After successful verification, the bank deposits the amount worth received e-cash to customer


Customer Panel

PAIR

Status: Paired

Merchant ID: 123XXX2112

Payment Amount: 12

CANCEL PAY

Customer

Merchant Panel

PAIR

Amount request

12

Status: Paired

REQUEST

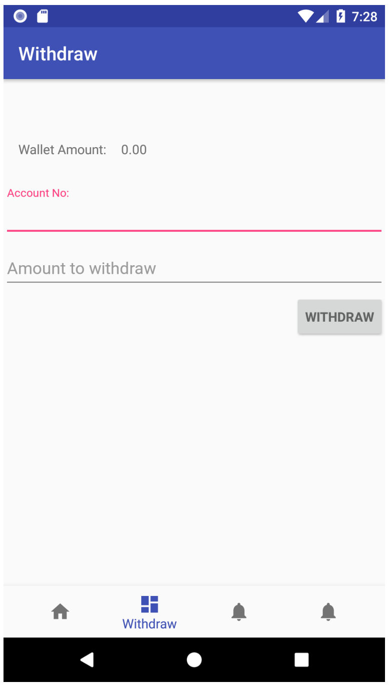
(a) Customer (b) Merchant

Figure 6.4: Customer and Merchant interfaces during payment process

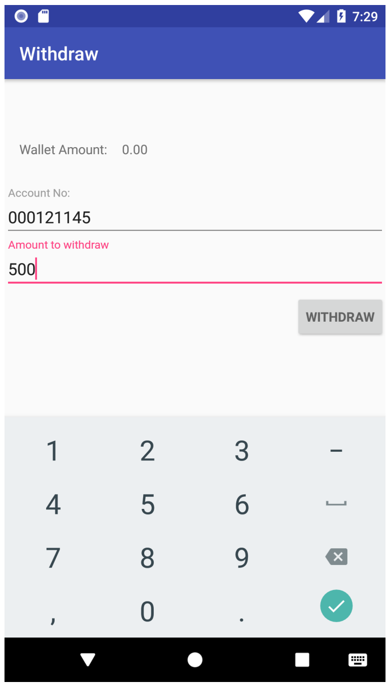
account.

1. Customer initiate the e-cash withdraw request to the bank with amount to withdraw and account to deposit.
2. Bank replies with random nonce (n).
3. Customer application sign the received nonce value and then send it to the bank with customer public key, signed value of public key and signed e-cash.
4. The bank verify the customer public key and then verify the nonce value and e-cash using verified public key of customer.
5. The bank deposit the received e-cash at customer account and store the transaction log.

Customer/ Merchant interfaces for withdrawal process are shown in the figure 6.5.



(a) Initial interface



(b) Filled Interface

Figure 6.5: Customer/ Merchant interfaces for withdrawal process

Client mobile application Android code segment for withdrawal process is given in appendix B.

6.5 Summary

This chapter shows the development of mobile application and server side implementation to prove the concept is possible to implement and use practically. Mobile development is using Andorid language and server side development is using PHP language. All the codes and interface designs are included with this chapter and with appendixes.

Chapter 7

Conclusions

Today, e-cash which is money in electronic format is using almost everywhere and every applications which are mostly depends on information systems, services or products. The applications of use of e-cash shape the way e-cash is issued, circulated and circulated. When all most all the information applications are heading to e-cash which is the electronic format of money it is important to develop new electronic payment schemes to focus on which properties are necessary and should be relaxed. This thesis has not only proposed a new offline micro payment scheme, but also mentioned a framework that should be considered when new micro payment scheme is being developed.

Electronic money or e-cash is a form of traditional money in electronic format but both traditional and electronic money has same properties and functionality. This thesis has list down and describe what properties should be with money and how those can be applicable to electronic format of the money. Most of the features can be applicable to when money comes with electronic format but some can be avoided in some level of features. It is not necessary to have all of the features and properties to be there when money comes from traditional format to electronic format. It is possible to deviate from some properties and enhance some drawback depending on the use case of the information system we use e-cash.

Although an ideal e-cash payment model has the capability to satisfy all the type of payments ranging from cents to thousands of dollars. But in day to day current real life environment, we use cash for small amount of purchasing or payment for services, credit or debit cards for larger payments and checks or EFT methods for payments which are over the limits of card payments.

Most of the time the decision for selecting the channel of payments depends on the cost of the transaction and the amount of payment. Therefore micropayments model heavily depends on the transaction cost due to the cost involving for transaction phase should be minimal compared to transaction amount. Most of the existing schemes for micropayments are involved communication cost although the transactions get declined. This proposed scheme almost does not belong to any of the direct or indirect transaction or communication cost while making a complete or incomplete transaction as it does not involve with third-party services and it works off-line mode.

Security of the proposed scheme is heavily depends on the asymmetric cryptography. At the beginning of the process, every clients get registered with the bank and bank can have their own policies for new customer registration. Clients may have to get registered with existing bank account details or verifying client mobile number using one time password through SMS. Only after a successful client verification, bank will provide the signature of client generated client's own public key. Client may able to top-up the client's wallet using bank account or different methods provided through bank channels. This self registration with top-up features increase the Usability of the payment scheme while holding the validity of the customers.

When a customer wants to make a payment, every time customer sign a nonce value received from merchant to protect the freshness of the transaction and share the public key with the signature of it to protect the authenticity of the customer. Merchant can verify all of these off-line and without involving direct or indirect cost. Therefore the merchant can anytime decline the transaction without losing a fraction of cents. If it is necessary and depending on the use case, it is possible to verify and authenticate the merchant using same method used for customer verification by merchant.

This proposed scheme is suitable not only between two mobile devices,but also capable to handle between a mobile device of customer and a static hardware equipment which can process and store required details. Due to the capability of handling transactions offline mode, it is possible to use any where any time without depending on other third party services availability.

The proposed scheme using ring signature is capable to handle e-cash anonymously and most of the use cases ignore the requirement of the traceable while dealing with micro payments. But depending on the requirement and with enhancements to some forms of ring signatures, malicious signer is traceable and can be revoked by the trusted authority.

These conclusions can be concluded and summarized as that the proposed cryptographically secured micropayment scheme, if implemented as proposed with the necessary requirements depending on a real life payment applications would be a successful micro payment scheme. The rationale in this statement is that the properties we discussed and satisfied heavily depend and focus on the driving forces for the development of electronic payments.

References

[1] 2013 online fraud report. `http://forms.cybersource.com/forms/fraudreport2013`. Accessed: 2017-09-30.

[2] Octopus. `http://www.octopus.com.hk`. Accessed: 2017-11-23.

[3] Paypal. `https://www.paypal.com`. Accessed: 2017-11-14.

[4] Paysafecard. `http://www.paysafecard.com`. Accessed: 2017-11-14.

[5] V. Ahuja. *Secure commerce on the Internet.* AP Professional Boston, 1997.

[6] S. T. Ali, D. Clarke, and P. McCorry. The nuts and bolts of micropayments: a survey. *arXiv preprint arXiv:1710.02964*, 2017.

[7] D. Blankenhorn. Charging for content, e-commerce times. *Online: www. ecommercetimes. com/perl/story/306. html*, 2001.

[8] K. Chaudhary, X. Dai, and J. Grundy. Experiences in developing a micropayment system for peer-to-peer networks. *International Journal of Information Technology and Web Engineering (IJITWE)*, 5(1):23–42, 2010.

[9] X. Dai and J. Grundy. Netpay: An off-line, decentralized micro-payment system for thin-client applications. *Electronic Commerce Research and Applications*, 6(1):91–101, 2007.

[10] X. Dai, J. Grundy, and B. W. Lo. Comparing and contrasting micropayment models for e-commerce systems. In *Info-tech and Info-net, 2001. Proceedings. ICII 2001-Beijing. 2001 International Conferences on*, volume 6, pages 35–41. IEEE, 2001.

[11] D. Geer. E-micropayments sweat the small stuff. *Computer*, 37(8):19–22, 2004.

[12] S. Ghosh, A. Majumder, J. Goswami, A. Kumar, S. P. Mohanty, and B. K. Bhattacharyya. Swing-pay: One card meets all user payment and identity needs: A digital card module using nfc and biometric authentication for peer-to-peer payment. *IEEE Consumer Electronics Magazine*, 6(1):82–93, 2017.

[13] R. Hauser, M. Steiner, and M. Waidner. *Micro-payments based on iKP.* IBM TJ Watson Research Center, 1996.

[14] M. Jain, S. Lal, and A. Mathuria. A survey of peer-to-peer micropayment schemes.

[15] D. Jayasinghe, K. Markantonakis, I. Gurulian, R. N. Akram, and K. Mayes. Extending emv tokenised payments to offline-environments. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 443–450. IEEE, 2016.

[16] J. Katz, A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC press, 1996.

[17] S. Kungpisdan, B. Srinivasan, and P. D. Le. A secure account-based mobile payment protocol. In *null*, page 35. IEEE, 2004.

[18] M. Lesk. Micropayments: An idea whose time has passed twice? *IEEE Security & Privacy*, 2(1):61–63, 2004.

[19] M. Lesk. Micropayments: An idea whose time has passed twice? *IEEE Security & Privacy*, 2(1):61–63, 2004.

[20] M. S. Manasse et al. The millicent protocols for electronic commerce. In *USENIX Workshop on Electronic Commerce*, 1995.

[21] T. Moore and N. Christin. Beware the middleman: Empirical analysis of bitcoin-exchange risk. In *International Conference on Financial Cryptography and Data Security*, pages 25–33. Springer, 2013.

[22] Y. Mu, V. Varadharajan, and Y.-X. Lin. New micropayment schemes based on pay words. In *Australasian Conference on Information Security and Privacy*, pages 283–293. Springer, 1997.

[23] A. Odlyzko. The case against micropayments. In *International Conference on Financial Cryptography*, pages 77–83. Springer, 2003.

[24] A. Odlyzko. Privacy, economics, and price discrimination on the internet. In *Proceedings of the 5th international conference on Electronic commerce*, pages 355–366. ACM, 2003.

[25] I. Papaefstathiou and C. Manifavas. Evaluation of micropayment transaction costs. *Journal of Electronic Commerce Research*, 5(2):99–113, 2004.

[26] R. Párhonyi, L. J. Nieuwenhuis, and A. Pras. Second generation micropayment systems: lessons learned. In *Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government*, pages 345–359. Springer, 2005.

[27] S. Poon and P. M. Swatman. An exploratory study of small business internet commerce issues. *Information & management*, 35(1):9–18, 1999.

[28] R. L. Rivest. Peppercoin micropayments. In *International Conference on Financial Cryptography*, pages 2–8. Springer, 2004.

[29] R. L. Rivest and A. Shamir. Payword and micromint: Two simple micropayment schemes. In *International Workshop on Security Protocols*, pages 69–87. Springer, 1996.

[30] A. Salomaa. *Public-key cryptography.* Springer Science & Business Media, 2013.

[31] M. Sirbu and J. D. Tygar. Netbill: An internet commerce system optimized for network-delivered services. *IEEE Personal Communications*, 2(4):34–39, 1995.

[32] K. Wei, A. J. Smith, Y.-F. Chen, and B. Vo. Whopay: A scalable and anonymous payment system for peer-to-peer environments. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 13–13. IEEE, 2006.

[33] B. Yang and H. Garcia-Molina. Ppay: micropayments for peer-to-peer systems. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 300–310. ACM, 2003.

Appendix A

PAT Analyser

PAT Analyser: registration.csp

```
//=======================Model Details=======================

enum{BankHello, BankHelloDone,BankFinished,HelloRequest,
     Certificate, Finished,Hello, Client, Bank, BankValue,
     BankNonce, signedNonce, Coins, GeneratedCertificate,
     BankNonceVerify, BankSignedClientCertificate };

//====================Channel Definitions====================

channel network 0;

//====================Process Definitions====================

ClientProc() = ClientRequest();
ClientRequest() =
   network!Client.Hello ->
   network?BankHello ->
   network?BankNonce ->
   network?BankHelloDone ->
   network!Client.GeneratedCertificate ->
   network!Client.signedNonce ->
   network!Client.Finished ->
   network?BankNonceVerify ->
   network?BankSignedClientCertificate ->
   network?BankFinished ->
   ClientRequest();
```

```
BankProc() = BankNegotiate();
BankNegotiate() =
  network?x.Hello ->
  network!BankHello ->
  network!BankNonce ->
  network!BankHelloDone ->
  network?x.GeneratedCertificate ->
  network?x.signedNonce ->
  network?x.Finished ->
  network!BankNonceVerify ->
  network!BankSignedClientCertificate ->
  network!BankFinished ->
  BankNegotiate();


aSystem = ClientProc() ||| BankProc();

//============================Asserstion Definitions====================

#assert aSystem deadlockfree;
```

PAT Analyser: obtainCash.csp

```
//=======================Model Details=======================

enum{BankHello, BankHelloDone,BankFinished,HelloRequest, RequestCoins,
       Certificate, Finished,Hello, Client, Bank, BankValue, BankNonce,
           signedNonce, BankSendCoins, Coins, CoinsVerify,
           BankCoinsVerify, BankCertificateVerify, BankNonceVerify };


//====================Channel Definitions====================

channel network 0;

//====================Process Definitions====================

ClientProc() = ClientRequest();
ClientRequest() =
  network!Client.Hello ->
  network?BankHello ->
```

```
   network!Client.RequestCoins ->
   network?BankNonce ->
   network!Client.Certificate ->
   network!Client.signedNonce ->
   network!Client.Finished ->
   network?BankCertificateVerify ->
   network?BankNonceVerify ->
   network?BankSendCoins ->
   network?BankFinished ->
   network!Client.CoinsVerify ->
   network!Client.Finished -> ClientRequest();

BankProc() = BankNegotiate();
BankNegotiate() =
   network?x.Hello ->
   network!BankHello ->
   network?x.RequestCoins ->
   network!BankNonce ->
   network?x.Certificate ->
   network?x.signedNonce ->
   network?x.Finished ->
   network!BankCertificateVerify ->
   network!BankNonceVerify ->
   network!BankSendCoins ->
   network!BankFinished ->
   network?x.CoinsVerify ->
   network?x.Finished ->
   BankNegotiate();


aSystem = ClientProc() ||| BankProc();

//============================Asserstion Definitions====================

#assert aSystem deadlockfree;
```

PAT Analyser: spendCash.csp

```
//======================Model Details======================


enum{MerchantHello, MerchantHelloDone,MerchantFinished,HelloRequest,
        Certificate, Finished,Hello, Client, Merchant, MerchantValue,
            MerchantNonce,
        signedNonce, Coins, MerchantCertificateVerify, MerchantNonceVerify,
            MerchantCoinsVerify };


//====================Channel Definitions====================

channel network 0;

//====================Process Definitions====================

ClientProc() = ClientRequest();
ClientRequest() =
   network!Client.Hello ->
   network?MerchantHello ->
   network?MerchantNonce ->
   network?MerchantValue ->
   network?MerchantHelloDone ->
   network!Client.Certificate ->
   network!Client.Coins ->
   network!Client.signedNonce ->
   network!Client.Finished ->
   network?MerchantCertificateVerify ->
   network?MerchantNonceVerify ->
   network?MerchantCoinsVerify ->
   network?MerchantFinished -> ClientRequest();

MerchantProc() = MerchantNegotiate();
MerchantNegotiate() =
   network?x.Hello ->
   network!MerchantHello ->
   network!MerchantNonce ->
   network!MerchantValue ->
   network!MerchantHelloDone ->
   network?x.Certificate ->
   network?x.Coins ->
   network?x.signedNonce ->
   network?x.Finished ->
   network!MerchantCertificateVerify ->
   network!MerchantNonceVerify ->
   network!MerchantCoinsVerify ->
   network!MerchantFinished ->
   MerchantNegotiate();
```

```
aSystem = ClientProc() ||| MerchantProc();

//============================Asserstion Definitions====================

#assert aSystem deadlockfree;
```

PAT Analyser: depositCash.csp

```
//=======================Model Details=======================

enum{BankHello, BankHelloDone,BankFinished,HelloRequest,
       Certificate, Finished,Hello, Merchant, Bank, BankValue, BankNonce,
           signedNonce, Coins, SignedCoins, RedeemRequest,
           BankCertificateVerify, BankNonceVerify, BankCoinsVerify };


//====================Channel Definitions====================

channel network 0;

//====================Process Definitions====================

MerchantProc() = MerchantRequest();
MerchantRequest() =
   network!Merchant.Hello ->
   network?BankHello ->
   network!Merchant.RedeemRequest ->
   network?BankNonce ->
   network!Merchant.Certificate ->
   network!Merchant.SignedCoins ->
   network!Merchant.signedNonce ->
   network!Merchant.Finished ->
   network?BankCertificateVerify ->
   network?BankNonceVerify ->
   network?BankCoinsVerify ->
   network?BankFinished -> MerchantRequest();

BankProc() = BankNegotiate();
BankNegotiate() =
   network?x.Hello ->
```

```
  network!BankHello ->
  network?x.RedeemRequest ->
  network!BankNonce ->
  network?x.Certificate ->
  network?x.SignedCoins ->
  network?x.signedNonce ->
  network?x.Finished ->
  network!BankCertificateVerify ->
  network!BankNonceVerify ->
  network!BankCoinsVerify ->
  network!BankFinished ->
  BankNegotiate();


aSystem = MerchantProc() ||| BankProc();

//===========================Asserstion Definitions====================

#assert aSystem deadlockfree;
```

Appendix B

Proof of Concept

PHP Development: Sign a value

```
$plaintext = $_GET['key'];

$privateKey = openssl_pkey_get_private('file://private.key');
openssl_sign($plaintext, $signature, $privateKey, OPENSSL_ALGO_SHA512);
openssl_free_key($privateKey);
echo bin2hex($signature);
```

PHP Development: verify a signature

```
$plaintext = $_GET['txt'];
$signature = $_GET['sign'];

$publicKey = openssl_pkey_get_public('file://public.key');

$v = openssl_verify ( $plaintext, hex2bin($signature), $publicKey,
    OPENSSL_ALGO_SHA512);
openssl_free_key($publicKey);
echo $v;
```

PHP Development: Bank side registration part 1

```
header('Content-type: application/json');
```

```
$wallet = $_GET['wallet'];
$name = $_GET['name'];

$nonce = generateRandomString();

include 'db.php';

if($result = $mysqli->query("INSERT INTO registration(wallet,name,nonce)
    values('".$wallet."','".$name."','".$nonce."')")){
        echo $nonce;
}

function generateRandomString($length = 10) {
    $characters =
        '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $charactersLength = strlen($characters);
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, $charactersLength - 1)];
    }
    return $randomString;
}
```

PHP Development: Bank side registration part 2

```
header('Content-type: application/json');
$wallet = $_GET['wallet'];
$signedNonce = $_GET['signedNonce'];
$clientPubKey = $_GET['clientPubKey'];

include 'db.php';

$mysqli = new mysqli("localhost", "offlinePayment", "offlinePayment",
    "maruka");

if ($result = $mysqli->query("SELECT nonce FROM registration where
    wallet=".$wallet)) {
        while($row = $result->fetch_array()){
                $nonce = $row['nonce'];);
        }
```

```
}


$publicKey = openssl_pkey_get_public($clientPubKey);

$v = openssl_verify ( $nonce, hex2bin($signedNonce), $publicKey,
    OPENSSL_ALGO_SHA512);
openssl_free_key($publicKey);
if($v == 1){
        $privateKey = openssl_pkey_get_private('file://private.key');
        openssl_sign($clientPubKey, $pubKeySign, $privateKey,
            OPENSSL_ALGO_SHA512);
        openssl_free_key($privateKey);

        if($result = $mysqli->query("UPDATE registration SET pubKeySign =
            '".$pubKeySign."' WHERE wallet='".$wallet."'")){
                echo $pubKeySign;
        } else {
                echo "Error";
        }
} else{
        echo "Error";
}
```

Android Development: Client side Registration process

```
protected Boolean doInBackground(Void... params) {

    bankFunctions bnkFun = new bankFunctions();
    nonce = bnkFun.registrationRequestToBank();
    try {
        KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
        keyStore.load(null);
        privateKey = (PrivateKey) keyStore.getKey("offlineClientKey", null);
        if (privateKey != null) {
            publicKey =
                keyStore.getCertificate("offlineClientKey").getPublicKey();
        } else{
            KeyPairGenerator keyPairGenerator =
                KeyPairGenerator.getInstance(
                KeyProperties.KEY_ALGORITHM_RSA, "AndroidKeyStore");
                keyPairGenerator.initialize(
                    new KeyGenParameterSpec.Builder(
                        "offlineClientKey",
                        KeyProperties.PURPOSE_SIGN)
                        .setDigests(KeyProperties.DIGEST_SHA256,
                            KeyProperties.DIGEST_SHA512)
                        .setSignaturePaddings(KeyProperties.SIGNATURE_PADDING_RSA_PSS)
                        .build());
            KeyPair keyPair = keyPairGenerator.generateKeyPair();
            privateKey = keyPair.getPrivate();
            publicKey = keyPair.getPublic();
        }

        System.out.println("OfflinePayment: client privKey " + privateKey);
        System.out.println("OfflinePayment: client publKey " + publicKey);

        signedNonce = sign(nonce, privateKey);
        System.out.println("OfflinePayment: " + signedNonce);

        signedPublickKey =
            bnkFun.sendSignedNonceWithPublicKeyToBank(signedNonce,
            publicKey);
        System.out.println("OfflinePayment: Signed PK " + signedPublickKey);
    } catch (Exception e) {
        System.err.println("OfflinePayment: " + e.getMessage());
        e.printStackTrace();
    }

    addpara();


    return true;
}

public long addpara() {
        MyPara mp = new MyPara(this);
        mp.createRecords("walletID", ""+mobile);
        return mp.createRecords("signedPK", signedPublickKey);
}
```

PHP Development: Bank side cash top-up

```
header('Content-type: application/json');
$wallet = $_GET['wallet'];
$qty = $_GET['qty'];


include 'db.php';

$privateKey = openssl_pkey_get_private('file://private.key');


$data = array();

echo $mysqli->error;
if (!empty($wallet) && !empty($qty)){
       for ($x = 1; $x <= $qty; $x++) {
           if($result = $mysqli->query("INSERT INTO cash(value,wallet)
               values(1,'".$wallet."')")){

               if ($result = $mysqli->query("SELECT * FROM cash where
                   id=".$mysqli->insert_id)) {
                       while($row = $result->fetch_array()){
                                  $coin = $row['id'] . "|" . $row['date'] .
                                      "|" . $row['value'];
                                  openssl_sign($coin, $signature,
                                      $privateKey, OPENSSL_ALGO_SHA512);
                                  $signature = bin2hex($signature);
                                  $dataA = array("coin"=>$coin,
                                      "signature"=>$signature);
                                  array_push($data,$dataA);
                          }
                  }
           }
       }
}


openssl_free_key($privateKey);

$result->close();
echo $mysqli->error;
$mysqli->close();

echo json_encode( $data );
```

Android Development: Client side Cash top-up

```
protected Boolean doInBackground(Void... params) {
    bankFunctions bnk = new bankFunctions();
    String nonce = bnk.topupRequestToBank();
    try {
        KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
        keyStore.load(null);

        KeyPair keyPair = new
            KeyPair(keyStore.getCertificate("offlineClientKey").getPublicKey(),
            (PrivateKey) keyStore.getKey("offlineClientKey", null));

        String signedNonce = sign(nonce,keyPair.getPrivate());
        ArrayMap arrayMap = bnk.topup(mMobile, mAmount, signedNonce);

        addCash(arrayMap);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return true;
}

public long addCash(ArrayMap amp) {
    MyCash mp = new MyCash(this);

    for (int i = 0; i < amp.size(); i++) {
        String cash = (String) amp.keyAt(i);
        String cashSignature = (String) amp.valueAt(i);
        mp.createRecords("",cash,cashSignature,"","");
    }
    return 1;
}
```

Android Development: cash spending customer initiation

```
protected Boolean doInBackground(Void... params) {

    externalPartyFunctions exf = new externalPartyFunctions();
    if(exf.inititaeBluetoothPair()) {
        String nonce = exf.payRequestToMerchant();
        try {
            KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
                    keyStore.load(null);

            KeyPair keyPair = new
                KeyPair(keyStore.getCertificate("offlineClientKey").getPublicKey(),
                (PrivateKey) keyStore.getKey("offlineClientKey", null));

            MyPara mp = new MyPara(CustomerActivity.this);
            Cursor cs = mp.getPara("signedPK");
            cs.moveToNext();
            String signedPK = cs.getString(1);

            String signedNonce = sign(nonce, keyPair.getPrivate());
            String amount =
                exf.sendSignedNonceWithPublicKeyToMerchant(signedNonce,
                keyPair.getPublic(), signedPK);
            ArrayMap cashFromCustomerWallet = getCash(amount);
            ArrayMap signValueOfCash = signCash(cashFromCustomerWallet,
                keyPair.getPrivate());

            if(exf.pay(cashFromCustomerWallet, signValueOfCash)){
                removeCash(cashFromCustomerWallet);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    return true;
}

public ArrayMap getCash(String amount) {
    MyCash mC = new MyCash(this);
    ArrayMap mp = new ArrayMap();

    Cursor cs = mC.selectRecords(amount);
    cs.moveToFirst();
    while(!cs.isAfterLast()) {
        mp.put(cs.getString(0) ,cs.getString(1));
        cs.moveToNext();
    }
    return mp;
}

public ArrayMap signCash(ArrayMap cashMap, PrivateKey pk) throws Exception
    {

    ArrayMap mp = new ArrayMap();

    for (int i = 0; i < cashMap.size(); i++) {
        String cash = (String) cashMap.keyAt(i);
        String cashCusSignature = sign((String) mp.valueAt(i), pk);
        mp.put(cash,cashCusSignature);
    }
    return mp;
}

public long removeCash(ArrayMap amp) {
    MyCash mC = new MyCash(this);

    for (int i = 0; i < amp.size(); i++) {
        String cash = (String) amp.keyAt(i);
        String cashSignature = (String) amp.valueAt(i);
        mC.updateRecords(cash);
    }

    return 1;

}
```

Android Development: cash spending merchant side

```
protected Boolean doInBackground(Void... params) {
    externalPartyFunctions exf = new externalPartyFunctions();
    if(exf.inititaeBluetoothPair()) {
        String nonce = exf.payRequestToMerchant();
        try {
            KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
            keyStore.load(null);
            externalPartyFunctions exp = new externalPartyFunctions();
            PublicKey cusPK = exp.getCusPublicKy();
            ArrayMap cashMap = exp.getReceivedcach();
            boolean cs = false;
            for (int i = 0; i < cashMap.size(); i++) {
                String cash = (String) cashMap.keyAt(i);
                String cashSignature = (String) cashMap.valueAt(i);
                cs =
                    verify(cash,cashSignature,keyStore.getCertificate("bankKey").getPublicKey()
                if(cs = false)
                        return false;
            }
            ArrayMap cusSignedCashMap = exp.getReceivedSignedcach();
            for (int i = 0; i < cusSignedCashMap.size(); i++) {
                String cash = (String) cusSignedCashMap.keyAt(i);
                String cashSignature = (String) cusSignedCashMap.valueAt(i);
                cs = verify(cash,cashSignature,cusPK);
                if(cs = false)
                    return false;
            }
            addCash(cashMap);
            addWallet(cusSignedCashMap);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    return true;
}

public boolean verify(String plainText, String signature, PublicKey
    publicKey) throws Exception {
    Signature publicSignature = Signature.getInstance("SHA256withRSA");
    publicSignature.initVerify(publicKey);
    publicSignature.update(plainText.getBytes(UTF_8));
    byte[] signatureBytes = Base64.getDecoder().decode(signature);
    return publicSignature.verify(signatureBytes);
}

public boolean addCash(ArrayMap amp) {
    MyCash mp = new MyCash(this);

    for (int i = 0; i < amp.size(); i++) {
        String cash = (String) amp.keyAt(i);
        String cashSignature = (String) amp.valueAt(i);
        mp.createRecords("",cash,cashSignature,"","");
    }

    return true;
}

public boolean addWallet(ArrayMap amp) {
    ReceivedCash mp = new ReceivedCash(this);

    for (int i = 0; i < amp.size(); i++) {
        String cash = (String) amp.keyAt(i);
        String cashSignature = (String) amp.valueAt(i);
        mp.createRecords("",cash,cashSignature);
    }

    return true;
}
```

Android Development: cash withdraw merchant side

```
protected Boolean doInBackground(Void... params) {
    bankFunctions bnk = new bankFunctions();
    String nonce = bnk.withdrawRequestToBank();
    try {
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance(
                KeyProperties.KEY_ALGORITHM_RSA, "AndroidKeyStore");
        keyPairGenerator.initialize(
                new KeyGenParameterSpec.Builder(
                        "offlineClientKey",
                        KeyProperties.PURPOSE_SIGN)
                        .setDigests(KeyProperties.DIGEST_SHA256,
                            KeyProperties.DIGEST_SHA512)
                            .setSignaturePaddings(KeyProperties.SIGNATURE_PADDING_RSA_PSS)
                        .build());
        KeyPair keyPair = keyPairGenerator.generateKeyPair();
        String signedNonce = sign(nonce,keyPair.getPrivate());
        ArrayMap wCash = getCash(mAmount);

        if(bnk.withdraw(mMobile, wCash, signedNonce)) {
            removeCash(wCash);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return true;
}

public long removeCash(ArrayMap amp) {
    MyCash mC = new MyCash(this);

    for (int i = 0; i < amp.size(); i++) {
        String cash = (String) amp.keyAt(i);
        String cashSignature = (String) amp.valueAt(i);
        mC.updateRecords(cash);
    }
    return 1;
}

public ArrayMap getCash(String amount) {
    MyCash mC = new MyCash(this);
    ArrayMap mp = new ArrayMap();

    Cursor cs = mC.selectRecords(amount);
    cs.moveToFirst();
    while(!cs.isAfterLast()) {
        mp.put(cs.getString(0) ,cs.getString(1));
        cs.moveToNext();
    }
    return mp;
}
```