ISSUES IN ELECTRONIC PAYMENT SYSTEMS: A NEW OFF-LINE TRANSFERABLE E-COIN SCHEME AND A NEW OFF-LINE E-CHECK SCHEME

BY

WONG HA YIN

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF PHILOSOPHY

IN INFORMATION ENGINEERING

AUGUST 2001

Abstract

With the onset of the Information Age, our nation is becoming increasingly dependent upon network communications. Electronic commerce becomes more and more important in our daily life. A key requirement for electronic commerce is the development of secure and efficient electronic payment systems.

In this thesis, we give an overview of the cryptography techniques for designing electronic payment systems. And the issues of security and privacy in off-line electronic payment systems are studied by introducing 6 basic requirements of the systems, discussing their history and classify system protocols according to their efficiency. We also study a simple and efficient single-term e-coin system in details

In particular, we will present two new results of electronic payment systems. One is a transferable e-cash scheme. Transfer the money from person to person without the involvement of a bank is a very important property in tradictional cash system, but there are only a little researcher talking about the transferability because of its difficulty of implementation. Our protocol is the third e-cash system that offers transferability. It is much more efficient compare with two old systems which use multiple terms.

The other payment system we presented in the thesis comes in a form as digital checks. We proposed two off-line e-check systems in this thesis. One is very simple but only supports partial anonymous. The other one is quite complex compare with previous one but it supports both privacy and untraceability in all transactions.

摘要

隨着資訊時代的到來,人類越來越依賴網路通訊。電子商貿在我們的日常生活裡變得越來越重要。因此建立一個保密和有效的電子付費系統是非常必需的。

在這一篇論文裡，我們概括了一些個可以用在電子付費系統的加密技術。通過介紹電子付費系統的基本要求，討論了系統的發展歷史並依照系統協議來分類，我們研究有關付費系統保安及用者私隱。而且我們詳細介紹了一個簡單及有效的單項電子貨幣系統。

在論文裡,我們提出了兩個新的電子付費系統設計，其中一個是可轉移的離線電子貨幣系統。在傳統的現金系統裡，金錢的轉讓是一個很重要的特性。但是直到現在只有少數研究人員討論過這個問題。因爲完成一個可以任意轉移的電子貨幣系統是非常難的。到目前爲止，我們的設計是第三份可以達到任意轉移目的的系統協議。和之前的兩個系統比較，我們的系統更加有效及簡單。

另外，我們設計了另一種電子付費系統 – 電子支票。在論文裡，我們一共建立兩個離線電子支票系統。其中一個非常簡單，但是只支持用者一部份的私隱。另一個系統相對的比較複雜，但它同時支持用者的私隱並且具有不可追蹤性。我們的設計更加完善了電子商貿的支付手段。

Acknowledge

This thesis would not have been possible without support of my advisor, friends and colleagues. I owe the warmest thanks to my academic supervisor Professor Victor K. W. Wei for his resourceful guidance on my research process. He initiated my involvement with the field of cryptography and provides constant support for my educational and professional requests.

I want to extend my thanks to the members in information Integrity Laboratory. Because of their psychological support, I have a very happy and privileged school life in these two years.

Finally, a heartfelt thank to my parents and my brother. Without their initiation I would not have started this degree.

Contents

Chapter 1

Introduction

The topic of this thesis, electronic payment system, is one of the fast-moving areas of computer science and mathematics research today. In these systems money is no longer represented by an object of trusted value (e.g. paper cash, gold, coins), but by information that is worth money. When the payment is effected across a computer network, it is called electronic payment.

We are about to step into a different realm. We'll still buy things in much the same way that we buy them today except that Net will just serve as the transaction ether. Under this situation, electronic payment systems have applications beyond replacing modem paper cash payment system. You can say that everywhere where value is assigned to an object by general trust or guarantee, there is the possibility to use electronic payment system.

In this chapter, we begin by quickly reviewing traditional payment forms and analyzing their shortcomings, to motivate the subsequent description of today's electronic payment system.

1.1 Traditional Payment Systems

Traditional payment systems can be divided into two groups: Payments by Cash and Payments by Instruction.

Traditional cash can be used spontaneously and instantaneously, to make payments from person to person without the involvement of a bank. Because of untraceability, cash payment system also offers privacy. The traditional cash is wide

acceptable for low and medium value purchases, which make up the bulk of our everyday transactions.

But traditional cash also has several shortcomings. It is easy to forged by today's high technology, notes can be copied using sophisticated color copier machines. Since cash comes in fixed denominations, many coins and notes need to change hands to pay a single amount. Coins are too heavy to carry around in bulk; both coins and notes are easily lost, stolen and destroyed. Because cash can be passed on many times without the need for settlement by a bank and without revealing user's identity, it increases the difficulty of governmental organizations to trace criminal money such as laundering, bribery and extortion. Moreover, since coins and bank notes payments need physical proximity of payer and payee, they cannot be used for payments over the phone or the Internet.

Payments by instruction can solve some of the problems of traditional cash system. During the payment instead of value itself, the payer transfers to the payee an instruction (consider checks and credit cards) directing the payer's bank to transfer a specified amount. Credit cards can be used over the phone or the Internet and the checks can be mailed by post. Because the actual values movement is done in the bank, the risks or theft and lose are largely overcome.

But payments by instruction also have many shortcomings. First of all, it doesn't support privacy, payee need to show his authenticity during the payment and the bank or central authority can trace all the payments not only the actual transfer of value but also user's identity. Most of instructions, such as credit cards and debit cards need on-line processing. An on-line payment can suffer from serious network delays or network breakdown. Although check payment system can be verified off-line, but it cannot readily issued by the banks and don't allow instantaneous payment.

1.2 Electronic Payment System

Any financial transaction involving the electronic transmission of information is

called *electronic commerce*. A particular kind of electronic commerce is that of *electronic payment*. An electronic payment protocol is a series of transactions, at the end of which a payment has been made, transmit a packet of digital information issued by a third party.

Compare with traditional payment systems, electronic payment systems have higher acceptability and more cost-effective for low-value purchases. It is possible to verify payments off-line with cross-platform portability. Moreover, electronic payments offer storage and transportation convenience while protecting users against loss, theft and accidental destruction. The electronic payment can make through the Internet or the phone, since physical proximity of payer and payee should not be needed. The electronic payment system also offers privacy and untraceability to its user. The account holders can hide transaction details from the bank and the confidential payment become possible. In additional to the privacy and untraceability, an electronic payment system can accept any special design to discourage criminal uses, such as laundering, bribery and extortion. With the right choice of cryptographic techniques, many or the entire properties listed above can be achieved.

As the properties of electronic payment are so attractive, a lot of electronic payment systems have been built up around us today. Just in CUHK, you will find you are using electronic payment system every day: Photocopy card is used for copying papers and notes, PPS is used for bill payment, Electronic token is used for cloth washing; Octopus card is used for transportation. Even buying a drink, you can also use the electronic payment system (figure 1.1).

Figure 1.1: Octopus Card Payment System used in drink machine

1.3 Thesis Organization

Chapter 2 overviews basic techniques in cryptography in order reader can fully understand digital payment systems.

In chapter 3 we discuss the basic properties of digital payment system in details. A basic model of electronic payment system is described and the development history of electronic payment system is also introduced.

Chapter 4 we explain one of the most efficient digital cash system, the Ferguson's single-term off-line coins. And in chapter 5, we present a method that bundle different denomination into the system and provide an efficient way to store the coins.

We develop a new e-coin system that adds transferability to Ferguson's single-term e-cash. Chapter 5 describes the implementation of the protocol in details

And in Chapter 6, we present another kind of electronic payment systems — e-check system. A simple, efficiency but only support partial privacy e-check system is described first. And then comes another e-check system, which includes basic requirements of basic payment system.

Chapter 2

Cryptographic Techniques

In real world, we always need some mechanisms to identify authority parties and users involved in a payment-related transaction. The often-used method is the application of a human signature to a document that will serve as the legal basis for the transaction. In digital world, we require the equivalent mechanisms to proves that someone is going to stand behind the value is encoded in the pile of bits.

The essential elements of these mechanisms can be replicated across computer networks through the use of cryptographic techniques. In addition, cryptography is useful in protecting against a wide variety of other attacks on the communications between two parties. In this chapter we give a basic introduction to the essential cryptographic techniques necessary to understand how electronic payment systems function. Readers who are already familiar with this material may skip ahead to chapter 3 and refer back as required. Details in mathematic can be referred to books [Sch96, Sti95]

2.1 Encryption and Decryption

Historically, the primary purpose of cryptography has been to keep the message's information hidden from adversaries. A process of disguising a human readable form message in order to hide its substance is called *encryption*. The original message is referred to in cryptographic terms as *plaintext* or *cleartext*, and the resulting message is referred to as *ciphertext*. The reverse process that takes ciphertext as input and restores

the original plaintext is called *decryption*.

A cryptographic algorithm is a mathematical function used for encryption and decryption. All modern encryption algorithms use a key, denoted by *EK*. Let *P* denotes plaintext and *C* denotes ciphertext.

The encryption function *E* operates on *P* to produce *C*:

$$E_{EK}(P) = C$$

Whereas, the decryption function *D* operates on *C* to produce *C* by using decryption key *DK*:

$$D_{DK}(C) = P$$

Cryptanalysis is a science of recovering the plaintext message without knowledge of the key. There are different forms of attacks on a cryptosystem, which are detailed described in [Sti95].

2.1.1 Symmetric Encryption

In symmetric encryption schemes, the encryption key is the same as the decryption key. Therefore, both parties to a communication must first possess a copy of a single secret key. Examples of symmetric encryption include the Data Encryption Standard (DES), triple DES and the International Data Encryption Algorithm (IDEA).

2.1.2 Asymmetric or Public-Key Encryption

In Asymmetric Encryption (or Public–Key Encryption), each person gets a pair of keys, called the *public key* and the *secret key*. The public key is published and widely distributed while the secret key is never revealed. The need for exchanging secret keys is eliminated as all communications only involve public keys. Examples include RSA and Elliptic Curve Cryptosystem (ECC).

2.2 RSA

The standard algorithm for implementing public-key cryptography can be used for both encryption and authentication and it is called the RSA algorithm. It was first introduced in 1978 by Rivest, Shamir and Adlemant [RSA78] while working at MIT.

RSA relies upon modulo arithmetic. Its security is based on the difficulty of factoring very large numbers. To encode a message using RSA, a user needs a pair of keys.

p,q: two large primes, for maximum security they should be of equal length.

n: the public key, the product of *p* and *q*.

e: the public key, a randomly chosen number which is less than *n* and relatively prime to *(p-1)(q-1)*.

d: the private key, the inverse of (*e* mod *(p-1)(q-1)*) such that (*ed*=1 mod *(p-1)(q-1)*).

Note that, the two large prime factors *p* and *q* must be kept secret or destroyed, since if anyone could factor *n* into *p* and *q*, the private key *e*, could be obtained.

Encryption is simple. A message, *M*, is first broken into a series of blocks and represent each block as an integer m_i. The encrypted message, *C*, will be made up of a similarly sized message blocks, c_i, of about the same length. The encryption formula is

$$c_i = m_i^e \bmod n$$

And the decryption formula is

$$m_i = c_i^d \bmod n$$

Thus with RSA, each owner of a key pair holds *d* secret, and issues *e* and *n* as her public key. The owner does not need to know about *p* and *q*, although knowledge of these factors may be used to speed up calculations. RSA gains its security from the difficulty of factoring large prime numbers. Recovering plaintext *M* from the corresponding ciphertext *C*, given the public key (*e, n*) is equivalent to the problem of factoring *n*, depends on *n*'s length.

2.3 Blind Signatures

Chaum first proposed blind signature in 1982 [Cha83], and constructed first blind signature protocol using RSA algorithm [Cha85]. Blind signature method is very important for implementing voting and digital cash protocol. The use of blind signature is a method for allowing a person to sign a message without being able to see its contents.

We can think the process of blind signature as a notary places a wax seal on the outside of an envelope, which contain a piece of paper that the notary doesn't know its content. When a blind digital signature is used, cryptography techniques replace the envelope and wax seal. The user enciphers the digital document, which is comparable to putting the document in an envelope. The notary places a digital signature on the encrypted document in the envelope just like a max seal to prove the authenticity of the document.

Mathematically, the blind signature protocol works as follows. Alice wants bank to blindly sign a message x. The bank has a public key e, private key d, and a public modulus $n = pq$.

1. Alice chooses a blinding factor r, and sends $xr^e \pmod{n}$ to the bank.
2. The bank signs it: $\left(xr^e\right)^d = rx^d \pmod{n}$.
3. Alice gets the blind signature by dividing out the blinding factor: $\left(rx^d\right)/r = x^d \pmod{n}$.

Since r is random, Alice's bank cannot determine x. Therefore, she cannot connect the signature with Alice's payment.

2.4 General Computation Protocols

In order to limit the affect of adversaries in a cryptosystem, Yao [Yao82] has given a general solution called *general computation protocols* (also called *fault-tolerant protocols* or *secure distributed computation protocols* or simply *cryptographic*

protocols). The protocol allows any set of parties to compute any function based on their secret inputs, such that every party knows the output of the function but the input of each party remains secret.

Although the protocol can be used to solve some electronic cash system problems, such as blind signature, it is extremely slow to implement because its security level depend on computational complexity. Therefore, general computation protocol is not applicable in practice.

2.5 Cut-and-Choose Method

Michael Rabin has used the cut-and-choose protocol for the first time in 1978. The reason for choosing the name "cut-and-choose" is its similarity to the classic protocol for dividing anything fairly.

1. Alice cuts the thing in half.
2. Bob chooses one of the halves for himself.
3. Alice takes the remaining half.

Alice has to divide fairly in step (1), because she doesn't know which half will bob choose in step (2).

2.6 Hash Functions

Hash functions play an important role in electronic cash systems. The hash function can compress its output, in such a way that none can efficiently find two inputs that compresses to the same string.

A hash function is a function f with at least following two properties [MOV96]

1. *compression — f maps an input x of arbitrary finite bitlength, to an output $f(x)$ of fixed bitlength n.*
2. *ease of computation — given f and an input x, $f(x)$ is easy to compute.*

It is generally assumed that the algorithmic specification of a hash function is public knowledge.

There are some different classes of hash function, here we just overview some of them which will be used in our payment systems.

A *one-way* hash function (OWHF) is a hash function f, which is easily compute $f(x)$, but it is computationally infeasible to

a. find an input x, give $y=f(x)$, and

b. find any second input which has the same output as any specified input, also given x, find $x' \neq x$ such that $f(x) = f(x')$.

A collision resistant hash function (CRHF) is a hash function f, which is computationally infeasible to

a. find any two distinct inputs x, x' which hash to the same output, such that $f(x)=f(x')$.

b. find any second input which has the same output as any specified input, also given x, find $x' \neq x$ such that $f(x)=f(x')$

2.7 Secret Sharing

The idea of secret sharing is to start with a secret, and divide it into pieces called shares which are distributed amongst users such that the pooled shares of specific subsets of users allow reconstruction of the original secret.

The simplest level of this scheme is to divide the message into n pieces, such that any t of them can be used to reconstruct the message. This scheme is called the (t, n)-threshold scheme.

Definition [MOV96]: *A (t, n) threshold scheme ($t \leq n$) is a method by which a trusted party computes secret shares S_i, $1 \leq i \leq n$ from an initial secret S, and securely distributes S_i to user P_i, such that the following is true: any t or more users who pool their shares may easily recover S, but any group knowing only t-1 or fewer shares may not. A perfect threshold scheme is a threshold scheme in which knowing only t-1 or fewer shares provide no advantage (no information about S whatsoever, in the information-theoretic sense) to an opponent over*

knowing no pieces.

There are several other algorithms in secret sharing, please refer to [MOV96] for details.

2.8 Zero-Knowledge Proofs

The phrase "zero knowledge" in the name means that any eavesdropper will gain zero knowledge even if he can overhear the entire conversation. Many people may know it as a *challenge and response* protocol in which one side comes up with a question and the other side offers the correct answer.

Zero-knowledge proofs are very useful in cryptography, particular in electronic payment systems. Proofs of identity (a party proves knowledge of his secret key without revealing it), knowledge of some information (e.g., the user proves knowledge of the coin's inner construction upon payment), or validity of encrypted messages (e.g., the use proves that he correctly encrypted his identity in an electronic coin) are clarifying examples.

Let us give a simple zero-knowledge protocol (ZKP) for explanation. The protocol is due to Chaum, Evertse, and Graff [CevdG88]:

1. The values of A, B and p are public and x is your secret. Alice want to show that she know x such that $A^x=B \bmod p$ without revealing x.
2. Bob sends Alice a random challenge b.
3. Alice sends back a response $s = r+ bx \bmod p\text{-}1$ and $h=A^r$ where r is a random number with $r < p$.
4. Bob verifies the response by finding that whether $A^s \bmod p$ equals to $hB^b \bmod p$. If it does not, then it is obvious that Alice don't know x.

In this case, $A^x \bmod p$ can be used as a digital signature of A if the value of x is kept secret. It is considered computationally infeasible to find this x given just A, B, and p.

2.9 Timestamps

Timestamps may be used to provide timeliness and uniqueness guarantees, to detect message replay. They may also be used to implement time-limited access privileges, and to detect forced delays.

The basic idea of binding a timestamp to a message is showed as follows. A trusted third party T (*the timestamp agent*) appends a timestamp t_1 to a submitted digital document D, or its hash value $f(D)$, signs the composite document, and returns the signed document including t_1 to the submitter. Subsequent verification of T's signature then is established, based on trust in T, the existence of the document at the time t_1. The timestamps must be secure to prevent adversarial resetting of a clock backwards so as to restore the validity of old messages, or setting a clock forward to prepare a message for some future point in time.

Chapter 3
Overview of Electronic Payment Systems

In this chapter we try to give a clear understanding of electronic payment system, as well as many concepts applicable to electronic payment systems in general, by showing their basic requirements, classifying them according to their efficient and introducing their history.

3.1 Life Cycle

In normal paper cash systems, there are three related parties: *bank*, *user* and *shop*. We will also use these terms for parties in electronic payment systems in this thesis.

We make a distinction between two kinds of electronic payment system: transferable, and non-transferable. Non-transferable electronic payment system is electronic money that has to be deposited with the bank, whenever it has been spent once. This is shown in the model of Figure 3.1. Transferable money can be withdrawn from the bank, then paid from user to user, and finally deposited by either a shop or a user. A transaction model for such a system is shown in Figure 3.2. Both systems use the same kind of transactions. Money is first withdrawn from the bank, then it is paid (either once or several times), and finally, it is deposited with the bank, which concludes the money's life cycle. The core of a cryptographic design of an electronic cash system then consists of the protocols for withdrawal, payment and deposit.

Depending on the functionality of the cash system, other cryptographic protocol may be need.

A remark one might make is that there is only one bank. Of course in some real-life situations there would be several banks. The more advanced electronic payment systems usually work fine with multiple banks. Just as with paper cash, there is a "Federal Bank" which "prints" all the money, and each bank does the verification of money for itself.

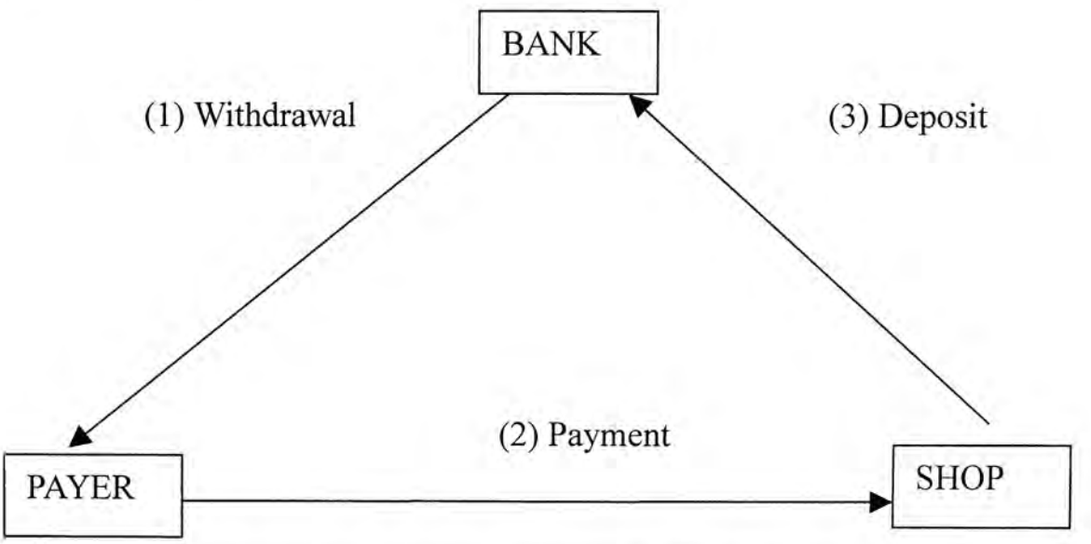

Figure 3.1: Life cycle of an non-transferable electronic payment system

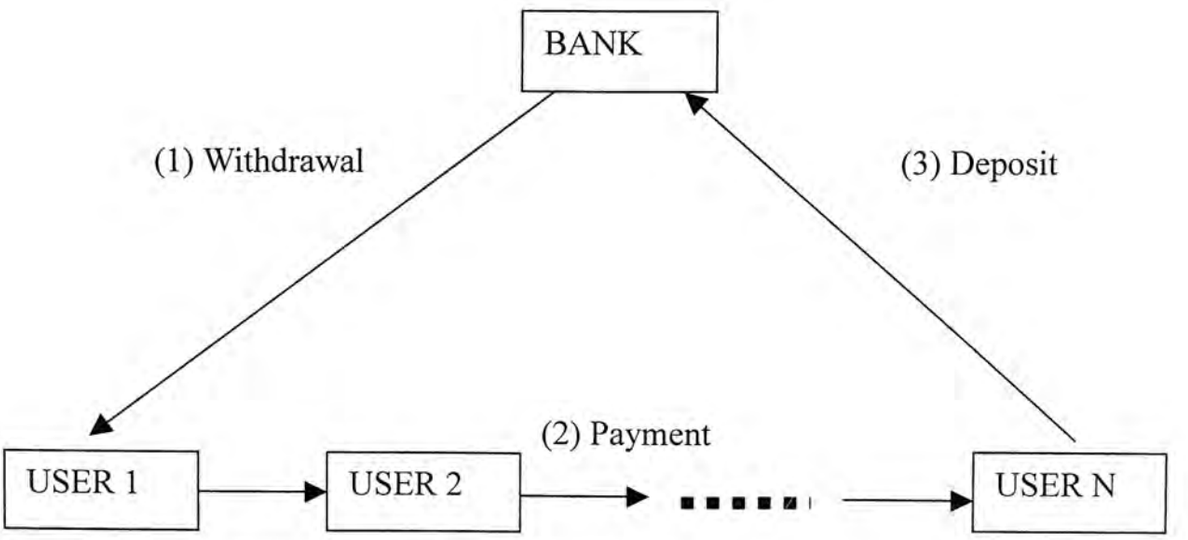

Figure 3.2: Life cycle of a transferable electronic payment system

3.2 Six Basic Requirements

Digital money is designed to construct an electronic payment system that modeled our daily paper cash payment system. Therefore digital money should fulfill all the requirements of paper cash. In 1991, two cryptographers Tatsuaki Okamoto and Kazua Ohta [OO92] formulated six requirements of the ideal electronic payment system.

Here we present in detail six basic requirements of digital money:

Independence (認錢不認人時地)

The security and use of digital money do not depend on any physical location, time and the user. The money can be transferred through computer networks into storage devices and vice versa.

Security (防僞)

Digital money cannot be counterfeited and double-spend. That is no one can create money without the intervention of the bank, or spend the same money more than once at different places.

Privacy (神不知鬼不覺)

The payment system should have protection against eavesdropping and the payer should be anonymous during the payment. The bank and the shop cannot trace the relationship between a person and a purchase.

Off-line Payment (兩造搞定)

Payment protocols can be implemented in either of two ways off-line or on-line. Tatsuaki Okamoto and Kazua Ohta believe that an ideal payment system should be working off-line. During a transaction, the only parties involved are the payer and the payee. No customer-bank or merchant-bank on-line connection is required.

Transferability (即時轉賬)

Transferability allows a user to spend money that he has received in a payment immediately without having to contact the bank. A payment is *transferable* if the payee can use the received money in other payment.

Divisibility (永遠有零錢)

Electronic money should have the ability to make "change" (some people like Okamoto believe that even the paper cash is undividable [OO92]). A quantity of digital money must be divisible into smaller amounts, and they must total up again when recombined. So users can always make an exact payment.

Implementing an electronic payment system that can fulfill all six requirements has a number of technical challenges. System designers always find a tradeoff between system complexity and features. Only a few of cryptographers have designed a system that meets all six requirements theoretically, but they have not been practice to date.

3.3 Efficiency

To implement a practical electronic payment system, efficiency is a very important issue. However, there may be many ways to determine whether an electronic payment system is "efficient" or not, since there is a great variance of efficiency requirements among different applications. Thus, instead of attempting to formally defined efficiency, we treat electronic payment system into 3 groups based on blind signature designs (following by Franklin and Yung [FY93]). Each group represents different efficiency level. These groups are informally described below:

E-cash based on general computation protocols

General computation protocols (described in section 2.4) can be used to implement blind signatures. User contributing one input (message to be signed), the bank contributing the other input (secret signature key), and the user alone receiving the output (signed message). Although general computation protocols are very powerful in that they allow arbitrary computation, they are extremely slow to implement. It is unreasonable for a coin scheme to have a communication cost that depends on the computational complexity of the underlying signature function algorithm. Will the bank mint a one-cent coin weigh fifty kilograms? A scheme of blind signature base on general computation protocols is secure, but definitely not efficient.

Cut-and-choose systems

Some e-cash systems have employed a cut-and-choose technique: A user presents $2n$ (where n is the security parameter) messages to the bank when he wants withdraw the money, the bank verify the correctness of n messages and blind signs the remaining messages. The cut-and-choose technique is a tool for a zero-knowledge proof of correctness of the coin, thus preserving user anonymity (see section 2.8 for a definition of zero-knowledge proof).

A very serious shortcoming of this straightforward implementation is that the probability of detecting a fraud increases only linearly in the number of message given, and thus a huge amount of data must be exchanged in order to achieve a sufficiently low probability of successful deception, i.e. the n is very large. Since the scheme's communication, computation and storage requirements are multiplied by a factor of n, the realization of cut-and-choose is still far from practical.

Single-term systems

"Single-term" means that there is only one "term" is used per coin. Generally it denotes all non cut-and-choose schemes those use n terms in transaction. In contrast to cut-and-choose based e-cash, recently proposed single-term systems employ efficient zero-knowledge (ZK) protocols providing the same functionality as cut-and-choose systems with a reduction by a factor of n in storage, computation and communication.

It avoids the expensive cut-and-choose method, by user blinding a secret key into coin or check; and the bank encodes user's identifier into the coin or check as the blinding-invariant part of the secret key. The invention of single-term systems makes the realization of an efficient off-line electronic payment system become possible.

This thesis focuses on single-term systems whose efficiency allows immediate practical application.

3.4 History

Although electronic payment systems become more and more important in our daily

life, public literature on the design of electronic payment systems is scarce and dispersed. There are several books have been published about electronic commerce, but these are all non-technical and focus on credit-card systems. In this section, we introduce the history of papers that are on cryptographic design of electronic payment systems, the main source for articles are come from the annual EUROCRYPT and CRYPTO conferences.

Early schemes for electronic payment systems used general computation protocols; including Damgård's on-line payment system [Dam90], Pfitzmnn and Waidner's off-line payment system [PW92]. Since most systems are so complex that they are notoriously slow to implement. But after the invention of blind signature techniques (definition in section 2.3) by Chaum, the real application of untraceable off-line electronic payment systems became possible. People studied the subject of electronic payment system and proposed various schemes. Most of them (e.g. Chaum, Fiat and Naor [CFN90], Okamoto and Ohta [OO90], Chan, Frankel and Tsiounis [CFT95, CFT96, CFT96]) used multiple challenges terms and "cut-and-choose" techniques in cash withdrawal as well as in payment phases in order to thwart cheating. Such kind of schemes are very inefficient since the communication, computation and storage requirements are multiplied by a factor n. Fianklin and Yung ([FY92], [FY93]) gave another construction for a provably secure coin scheme using a secret sharing line instead of multiple challenge terms (their schemes still used cut-and-choose in withdrawal phase). In 1993, three efficient "single-term" papers (Brand [Bra93], Cramer and Pedersen [CraPe93], Ferguson [Fer93]) have been proposed respectively, instead of cut-and-choose method, they used a direct construction.

Chapter 4
Ferguson's Single-term Off-Line Coins

In this chapter, we explain a single-term off-line coins system developed by Niels Ferguson [Fer93] in details. He presented a new construction for off-line electronic coins that is both far more efficient and much simpler than previous systems. The system only used a single term in random challenge and response method. And does not use a cut-and-choose methodology in withdrawal protocol, but used a direct construction.

We introduce base assumption and tools used by Ferguson's system first in section 4.1. And then explain the whole system in details in section 4.2.

4.1 Basic Assumption and Tools

4.1.1 Secure Hash Function

One-way and collision resistant hash functions are very useful in electronic payment system. They have many applications; one of them is the most important for system in this paper. If a protocol uses a number y somewhere, and if the protocol can be attacked with y chosen to have some special form, then y can be taken $y=f(x)$. This disables any user to come up with specially formed y. Later, we will show how the

hash function is applied in our payment systems.

Currently in whole paper, we are using Secure Hash Algorithm (SHA-1) (detail algorithms please refer to [MOV96]) as our cryptographic hash function for the electronic payment system, since it is both one-way and collision-intractable.

4.1.2 Polynomial Secret Sharing Scheme

The most difficult fraud to counter in electronic cash system is double-spending, i.e. user can spend same cash twice. For off-line payment system, we cannot detect this fraud immediately during the payment. The solution is we identify the double-spender after the fact. At each payment the user is required to release information in response to a challenge from the shop. One such release provides no clue to the user's identity, but two such releases are sufficient to identify the user uniquely. Employing Secret Sharing scheme (refer to section 2.7 for details) in challenge and response protocol can make this solution possible.

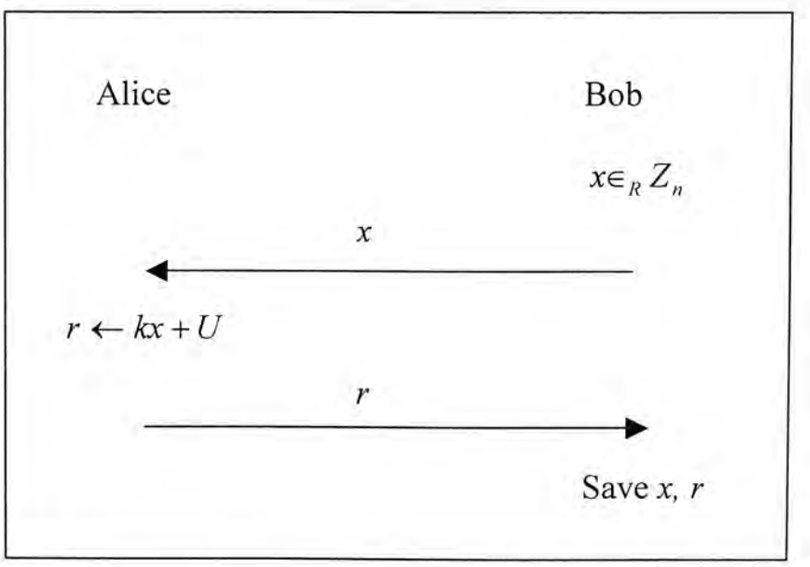

Figure 4.1: Shamir's Polynomial Secret Sharing Scheme

In protocol, Shamir's polynomial secret sharing scheme [Sha79] is used for double-spending detection. The scheme is based on polynomial interpolation, and the fact that a univariate polynomial $y = f(x)$ of degree t-1 is uniquely defined by t points

(x_i, y_i) with distinct x_i (since these define t linearly independent equations in t unknowns). To detect double-spending, we need to identify the user if we get two pairs of challenge and response, hence the polynomial secret sharing scheme can apply in the protocol (fingure 4.1).

Alice has her identity U and a secret number k.

1. Bob random generates a challenge x and send it to Alice
2. Alice response $r=kx+U$ to Bob.
3. Bob save x and r for later detection.

From above scheme, once challenge and response process will not show Alice's identity, but two such will identify Alice by solving two equations with two variables providing that two challenges are different. Later we will show how to use Blank Signature to prevents Alice from changing the values of k and U and use one way hash function to prevents Bob from generate two same challenges.

4.1.3 Randomized Blind Signature

In order to keep user's privacy in all transactions, we should use blind signature (see section 2.3) to prevent bank or shop has the ability to trace user's identity. In protocol, we use a blind signature call randomize blind signature that was proposed by Chaum in 1992 [Cha92]. The bank publics a key pair (v, n), a one-way function f, and a random number $g \in_R Z_n^*$.

Step 1 The user chooses a random number $a_1 \in Z_n^*$, and two blinding factors $\sigma \in_R Z_n^*$ and $\gamma \in_R Z_v$. She computers $\gamma^v a_1 g^\sigma$ and send it to the bank.

Step 2 The bank chooses its own contribution a_2 and sends it back to User.

Step 3 The user computes $f(a_1 \cdot a_2) - \sigma$, and sends the result to bank.

Step 4 The bank multiplies $\gamma^v a_1 g^\sigma$ by a_2 and $g^{f(a_1 \cdot a_2)-\sigma}$ to get $\gamma^v a_1 a_2 g^{f(a_1 a_2)}$. The bank computes the v'th root of this number and sends it to User.

Step 5 The user divides out the blinding factor γ to get the pair $\left(a, \left(a g^{f(a)}\right)^{1/v}\right)$. The

number $a = a_1 a_2$ is called the base number of the signature.

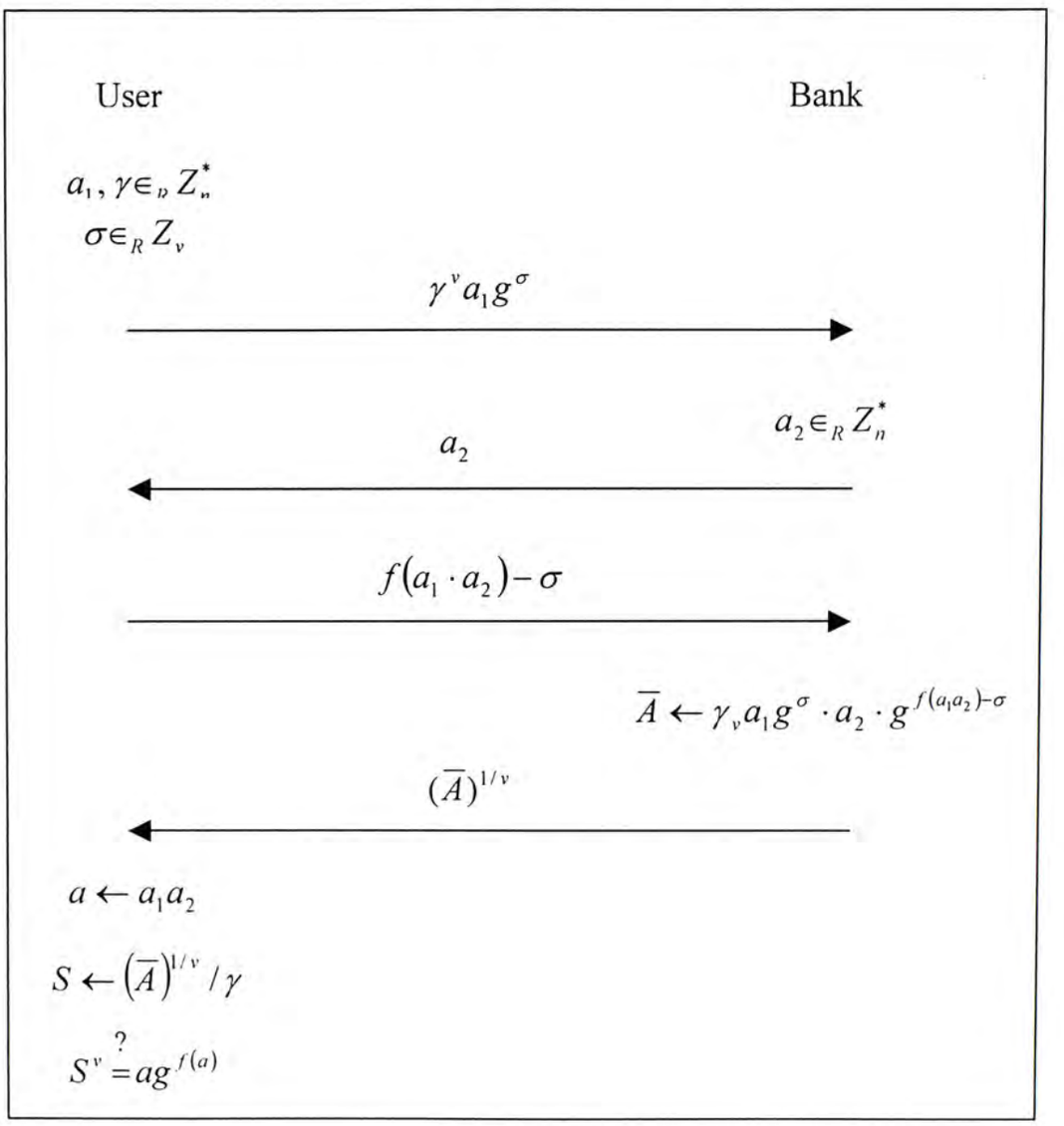

Figure 4.2: Randomized Blind Signature

This signature satisfies all the requirements of the proposed protocol.

1. **User cannot create signature herself**: *It is computationally infeasible to forge a signature pair of the form* $\left(a, \left(a g^{f(a)}\right)^{1/v}\right)$. Suppose the user wants to forge a signature pair (a, A), she should solve the equation $A^v = a g^{f(a)}$. The first way is fix A and try different values of a until you get lucky. The probability of success is very small equal to $1/v$. The second way is to choose

$ag^{f(a)}$ and compute out the values of A. This method need find the v'th root, which is as difficult as breaking RSA-key algorithm. No general methods are currently known that attempt to break RSA in this way.

If User has a large number of valid signature pairs, can she find a new one from exiting signatures? The answer is no because of one-way hash function f. User is difficult to get any valid signature by any combination of exiting signatures.

2. **The number of signature was randomly chosen**: The base number of signature a is the result of $f(a_1 \cdot a_2)$, where a_1 is contribution given by the user and a_2 is contribution given by the bank. This method use two random numbers make sure that the message sign by bank is randomly chosen.
3. **The bank receives no information regarding which signature User gets**: In the protocol, there are two blinding factors γ and σ that was randomly generated by the user. If the bank want to know what signature pair the user gets, she should find the value of a or $ag^{f(a)}$. Let's check all the communication to and from the bank, plus all the random choices that the bank made. He only gets the value of a_2, $f(a)$ and a_1 are all hidden by two blinding factors. From the bank's point of view, all possible signature pairs are equally likely.

4.2 The Basic Signal-term Cash System

This system uses secret sharing techniques. The blind signature used here is called a randomized blind RSA-based signature. In this case the bank still doesn't know what is signing, but she knows that the data wasn't chosen maliciously.

This system needs 3 numbers, C, A, and B. These will be of the form

$$C \leftarrow c g_c^{f(h_c^c)}$$
$$A \leftarrow a g_a^{f(a)}$$
$$B \leftarrow b g_b^{f(h_b^b)}$$

And two RSA-signatures from the bank:

$$(C' A)^{1/v} \text{ and } (C^U B)^{1/v}$$

Where the numbers g_c, g_a and g_b are publicly known and of the large order in group Z_n^*. The number h_c and h_b are elements of order n and f is a one-way function. U is the user's identity, v is the bank's public key.

4.2.1 The Withdrawal Protocol

The withdrawal protocol consists of three parallel runs of the randomized blind signature scheme. Two of the runs are the restricted version, and one is the unrestricted version.

Step 1 The user chooses randomly a_1, c_1, b_1 (her contributions to the base numbers) $\sigma, r, \phi, \alpha, \beta$ *and* γ (the blinding factors). The user computes $\gamma^v c_1 g_c^{\alpha}$, $\alpha^v a_1 g_a^{r}$, $\beta^v b_1 g_b^{\phi}$ and sends them to the bank.

Step 2 The bank then chooses her three contributions to the base numbers c_2, a_2, b_2. And sends $h_c^{c_2}, h_b^{b_2}$ and a_2 to the user. Sending a_2 directly allows the user to raise one of the resulting signatures to a power she chooses.

Step 3 The user chooses a random t_1 and computes

$$e_c \leftarrow f\left(h_c^{c_1 c_2}\right) - \sigma$$
$$e_b \leftarrow f\left(h_b^{b_1 b_2}\right) - \phi$$
$$a \leftarrow \left(a_1 a_2 \cdot f_2(e_c, e_b)\right)^{t_1}, \text{ where } f_2 \text{ is an one - way function.}$$
$$e_a \leftarrow \frac{1}{t_1} f(a) - r.$$

The user sends e_a, e_b and e_c to the bank.

Step 4 The bank compute the blinded version of A, B and C.

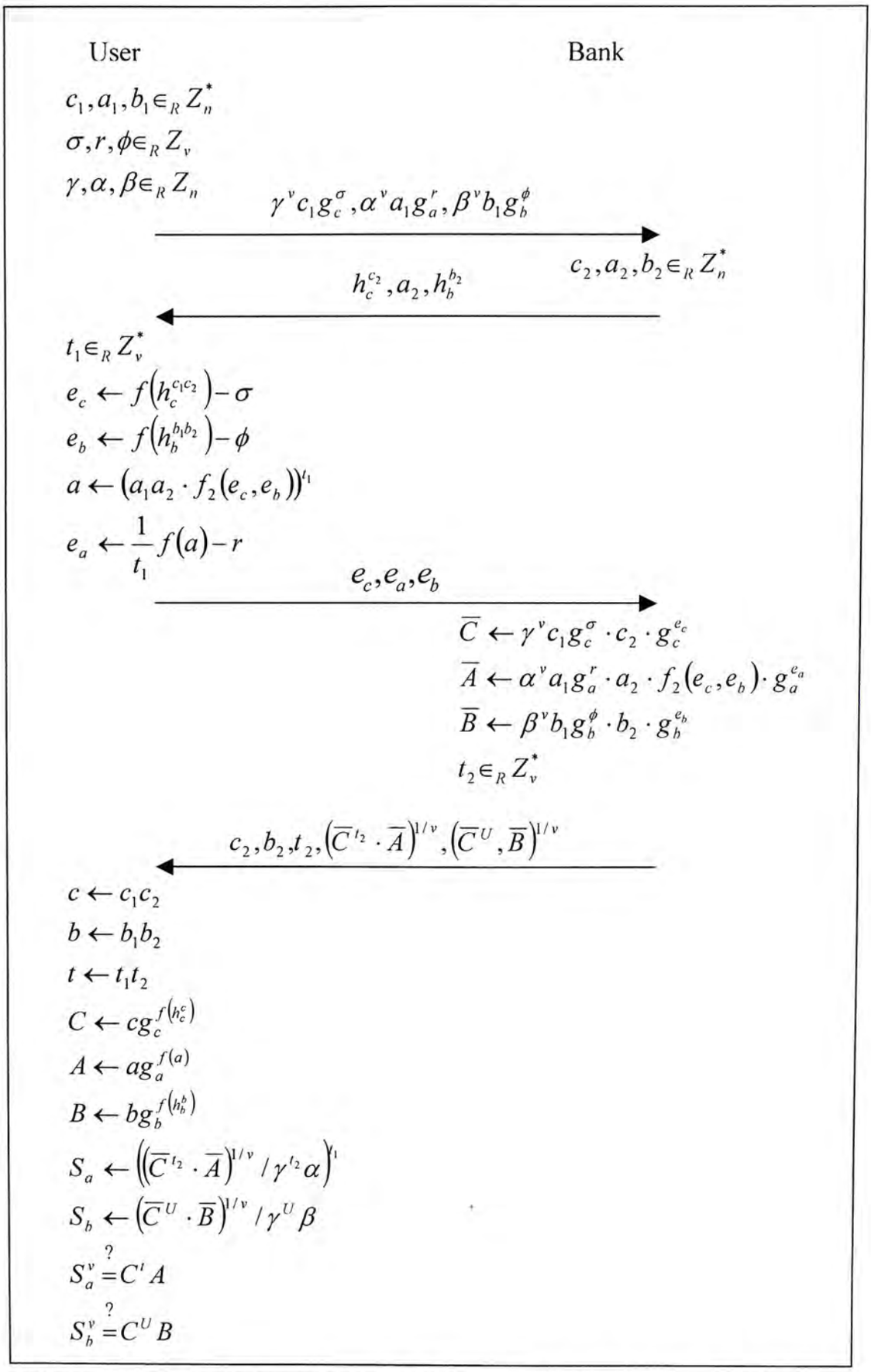

Figure 4.3: The withdrawal protocol of Ferguson's Single term e-cash

$\overline{C} \leftarrow \gamma^v c_1 g_c^{\sigma} \cdot c_2 \cdot g_c^{e_c}$, for $c = c_1 c_2$

$\overline{B} \leftarrow \beta^v b_1 g_b^{\phi} \cdot b_2 \cdot g_b^{e_b}$, for $b = b_1 b_2$

$\overline{A} \leftarrow \alpha^v a_1 g_a^{r} \cdot a_2 \cdot f_2(e_c, e_b) \cdot g_a^{e_a} = \alpha^v a g_a^{(1/t_1) f(a)}$

The following relations hold between the blinded numbers and their unblended values:

$\overline{C} = \gamma^v C, \overline{A} = \alpha^v A^{1/t_1}, \overline{B} = \beta^v B$

The bank then chooses a random t_2 and sends $c_2, b_2, t_2, \left(\overline{C}^{t_2} \cdot \overline{A}\right)^{1/v}$, and $\left(\overline{C}^{U}, \overline{B}\right)^{1/v}$ to the user.

Step 5 The user now constructs the numbers

$C \leftarrow c g_c^{f(h_c^c)}$, $A \leftarrow a g_a^{f(a)}$, $B \leftarrow b g_b^{f(h_b^b)}$

User computes two signatures

$S_a \leftarrow \left(\left(\overline{C}^{t_2} \cdot \overline{A}\right)^{1/v} / \gamma^{t_2} \alpha\right)^1$, $S_b \leftarrow \left(\overline{C}^{U} \cdot \overline{B}\right)^{1/v} / \gamma^U \beta$

The user checks the signatures she received are correct by verifying that $S_a^v = C^t A$ and $S_b^v = C^U B$ where $t = t_1 t_2$.

User ends up with the following numbers: *c, a, b,* (the base numbers) *t,* (the random parameter for the secret sharing line), S_a and S_b (the signatures). These 6 numbers plus the identity *U* are used as input to the payment protocol.

4.2.2 The Payment Protocol

Instead of using many challenge terms for double-spending detection, the protocol uses a polynomial secret sharing scheme [detail please refer to section 4.1.2]. The protocol (Figure 4.4) is described as below:

Step 1 The user sends *a, b,* and *c* to Shop.

Step 2 The shop replies with a randomly chosen challenge *x*.

Step 3 The user responses with $r=kx+U$ and the signature $\left(C^r A^x B\right)^{1/v}$ to the shop.

Step 4 The shop verifies the consistency of these two responses and then he accepts

the payment.

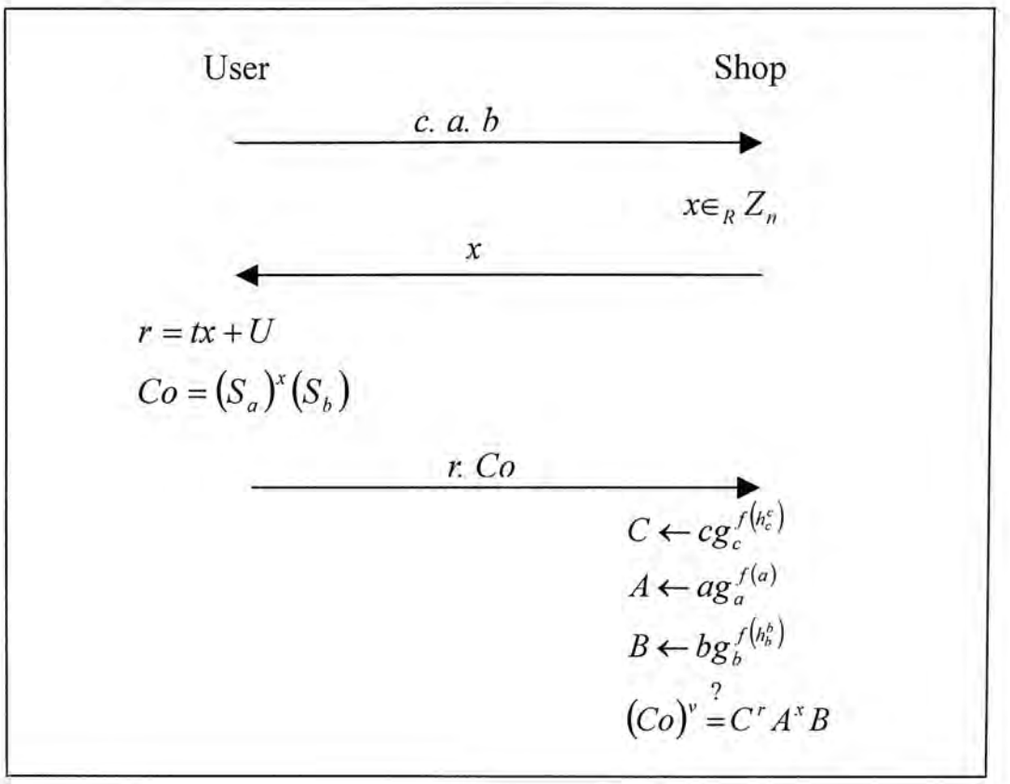

Figure 4.4: The payment protocol of Ferguson's single-term e-cash

4.2.3 The Deposit Protocol

At the end of the day, the shop will deposit the money to her account.

Step 1 The shop sends *c, a, b*, the challenge *x*, the responses *r* and the signature *Co* to the bank.

Step 2 The bank verifies the correctness of the coin and credits the shop with corresponding amount.

If the user spends the same coin twice, she must reveal two different points on the line $r \leftarrow tx + U$ which immediately allows the bank to determine her identity U.

Chapter 5
Cash with Different Denominations

In this section, we review a method of denomination bundling by [Cha90] and present an extension by observing an equivalent related to RSA number system.

5.1 Denomination Bundling

Now let's look at how we can bundle denomination terms into an e-coin. Instead of single key, the bank publics a list of keys presented different denomination of the coins. Let k denotes the number of different denominations, and $(d_1, d_2, \ldots d_k)$ denote a list of denominations. Let $(v_{d_1}, v_{d_2}, \ldots, v_{d_k})$ denote a list of valid RSA exponents for corresponding denomination, they are pairwise co-prime numbers. See Table 4.1 as example, in which each v_{d_i} is a prime number. The distribution of the coin denominations is chosen in such a way that it is guaranteed that no matter how the money is spent – at least a certain amount of money can be made in payment.

i	1	2	3	4	5	6	7	8	9
d_i	\$0.01	\$0.02	\$0.05	\$0.10	\$0.20	\$0.50	\$1.00	\$2.00	\$5.00
v_{d_i}	5	7	11	13	17	19	23	29	31

Table 5.1. A list of public keys present for different denomination

The withdraw protocol of the single-term e-coin system will not change except that the bank uses corresponding private-key to sign the coins depend on its denomination. The protocol can be executed in parallel as many times as required to withdraw the desired amount. In previous protocol, all random numbers of exponent are generated from group Z_v in order to make power method simpler but decrease the security level of the protocol. In our new protocol, apart from the random secret number a_1, a_2, b_1, b_2, c_1, c_2 are members of the set Z_n^*, the user also picks the random binding factors σ, r, ϕ, γ, α, β in the same set to make sure that no one can find the random numbers with high probability of success.

First, there is the fact that we use not just one, but a number of public exponents with the same RSA modulus to encode the difference denominations within a coinage. This raises the question whether, for instance, the user finds a special way to construct coins with valid format and denomination those are not deviating from the prescribed protocol. Recall that even if the user has a large number of valid signature pairs, it is still difficult for her to find new ones (details please refer to section 4.4.3). But can the user construct a coin of higher denomination from coin of lower denomination? More formally, the question is whether for some $a, b, c \in Z_n^*$ and two co-prime number $v_{i+1}, v_i \in Z_n^*$ $(where v_{i+1} > v_i)$, it is feasible to compute say, $\left(C^t A\right)^{1/v_{k+i}}$, $\left(C^U B\right)^{1/v_{k+i}}$ from the values $S_1 = \left(C^t A\right)^{1/v_k}$, $S_2 = \left(C^U B\right)^{1/v_k}$. We can show that this question is answered in the negative. It is equal to compute $(S_1)^{v_i/v_{i+1}}$ and $(S_2)^{v_i/v_{i+1}}$, the computation is just as difficult as finding RSA roots if V_i is not divisible by V_{i+1}.

5.2 Coin Storage

By using the way described previously to bundle the denomination into e-coin, it comes the other advantage that coin can be stored in a very small space. In this system great efficiency is derived from the fact that in the RSA crypto system signatures with different exponent can be combine together by multiplication and later separated.

Throughout the paper, all the arithmetic is performed in a RSA groups Z_n with n is the product of two (or more) distinct primes p and q. We find that given two distinct numbers x and y, and given two roots a and b which are relative prime. Then if we know a number s, which is equal to $x^{1/a} \cdot y^{1/b}$, we can easily find $x^{1/a}$ *and* $y^{1/b}$ separately.

The way to separate two roots is show as below:

Because *(a, b)=1*, there is a number $\bar{b}$, $0 \le \bar{b} < a$ with

$$b\bar{b} \equiv 1 \mod a$$
$$t = \lfloor b\bar{b} / a \rfloor$$

We can compute two numbers

$$c \leftarrow \frac{(s)^{b\bar{b}}}{x^t y^{\bar{b}}}$$
$$d \leftarrow s / x^{1/a}$$

where $c = x^{1/a}$ and $d = y^{1/b}$

How can this way apply to more than two terms? Suppose you want to separate

$$x_1^{1/a_1} \cdot x_2^{1/a_2} \cdot \cdots \cdot x_k^{1/a_k} \qquad (4.1)$$

for some $a_1, a_2, \ldots, a_k$, Please note that each a_i and $\phi(n)$ are pairwise co-prime. Define

$$A = \prod_{i=2}^{k} a_i .$$

Compute

$$x_2^{A/a_2} \cdot \cdots \cdot x_k^{A/a_k}$$

And rewrite equation (4.1)

$$x^{1/a_1} \cdot \left(x_2^{A/a_2} \cdot \cdots \cdot x_k^{A/a_k}\right)^{1/A}$$

We can apply the way described previously to separate x_1^{1/a_1}. Repeat this until all terms are separated.

For coins storage, we can just multiply the coins with different denomination together and easily recover the original signature by using the method described above. In original e-cash system (described in chapter 4) each coin need be store in about 250 bytes [Fer93]. With bundling different denomination into e-coin, the minimum coin size can be reduced to about 30 bytes. We can save several coins of the form $\{LA, LB, LC, S1, S2\}$ where $S1 \leftarrow \prod_{i=1}^{k} \left(C^{t_i} A\right)^{1/v_i}$, $S2 \leftarrow \prod_{i=1}^{k} \left(C^{U_i} B\right)^{1/v_i}$, $LA = a_1 \| a_2 \| \ldots \| a_k$, $LB = b_1 \| b_2 \| \ldots \| b_k$, $LC = c_1 \| c_2 \| \ldots \| c_k$. Please note that the exponents of each coin should be different.

With today's hard drives and memory chips there is absolutely no problem of storing any sensible number of coins.

Chapter 6
An Off-Line Transferable E-coin System

6.1 Introduction

Six criteria of an ideal cash system described by Tatsuaki Okamoto and Kazua Ohta are (1) independence, (2) security, (3) privacy, (4) off-line payment, (5) divisibility and (6) transferability. Cash protocols described in early papers have solved the first five characteristics. The last characteristic is the most interesting one considered in this chapter.

Transferability of electronic cash means that the payee in one payment transaction can spend the received money in a later payment to a third person without contacting the bank or another central authority between the two transactions. This property is only the issue for the off-line payment systems, since in on-line system, payee need communication with bank or a central authority to prevent double-spending during the payment transaction.

The ability to transfer the money on modern payment system is a basic requirement. Will you go to bank every time after you receive a value of money? It will not happen in our real life since we can transfer the paper cash to others freely. In non-transferable electronic cash system, deposit procedure after every transaction will extremely increase bank's workload and network traffic as more than billions of

payments are made around the world each day.

Although the ability to transfer paper money is very important in our daily life, there are only a few number of papers paid attention to this property. Until now, there are only two off-line electronic payment systems claimed that they could support transferability.

Okamoto and Ohta ([OO90] and [OO92]) proposed the first transferable payment system, which is also the first system that supports all 6 basic requirements. The scheme is extremely inefficient, since it used "cut-and-choose" techniques in cash withdrawal and multiple terms for challenge and response method in payment. After each transfer, both computational power and the size of cash will increase a lot. The protocol is so complex and difficult to understand (the concise description of the protocol in [OO90] requires 13 pages) that it is difficult to implement in the real world.

In [VA90], the writer presented the other transferable payment system that can pay a variable amount of money with a single check. The system protocol didn't use cut-and-choose protocol in withdrawal, but used a direct construction. However, it still use multiple terms in its payment protocol. Each user need to check n more signatures (where n is the number of terms), after each transfer, the computational power linearly increase by a factor n.

In this chapter, we present a new transferable off-line e-cash system motivated by Ferguson's withdrawal protocol. As the knowledge of the authors, the payment system is the third transferable e-cash system that has been proposed. The new construction is both far more efficient and much simpler than previous systems. It use single-term challenge and respond method instead of multiple terms, and use a direct method in withdrawal instead of cut-and-choose methodology. The computational power and storage space is so small that even the smart card can handle the transaction.

Except for transferability, we also show a simple mechanism that combine coins together to produces a new coin have a value, which is the sum of the values of all combined coins.

6.2 The Withdrawal Protocol

The basic concept to realize transferability is the idea of coin with no denomination, i.e. coin with no values. We have introduced a way to bundle denomination into e-coin in chapter 5, it is easy to identify a coin with no values, just assign one more public exponent present zero denomination. Table 6.1 is a modified denomination distribution table compare with table 5.1. Of course, users can obtained these kinds of zero value coins from the bank freely.

i	0	1	2	3	4	5	6	7	8	9
d_i	\$0.00	\$0.01	\$0.02	\$0.05	\$0.10	\$0.20	\$0.50	\$1.00	\$2.00	\$5.00
v_{d_i}	3	5	7	11	13	17	19	23	29	31

Table 6.1. A list of public keys present for different denomination

The withdrawal protocol is the same as Ferguson's withdrawal protocol as it meets all requirements of our system. Readers who are already familiar with Ferguson's withdrawal protocol may skip ahead to the payment protocol and refer back as required.

Before the execution of withdrawal protocol (Figure 6.1), there is an agreement of money amount between the bank and the user. Let's denote the amount of money is d and the corresponding public key is v.

Step 1 The user chooses 9 random numbers a_1, c_1, b_1, $\sigma, r, \phi, \alpha, \beta$, γ and sends $\gamma^v c_1 g_c^{\alpha}$, $\alpha^v a_1 g_a^r$, $\beta^v b_1 g_b^{\phi}$ to the bank.

Step 2 The bank then chooses 3 random numbers c_2, a_2, b_2 and sends $h_c^{c_2}, h_b^{b_2}$ and a_2 to the user.

Step 3 The user chooses a random number t_1 to compute

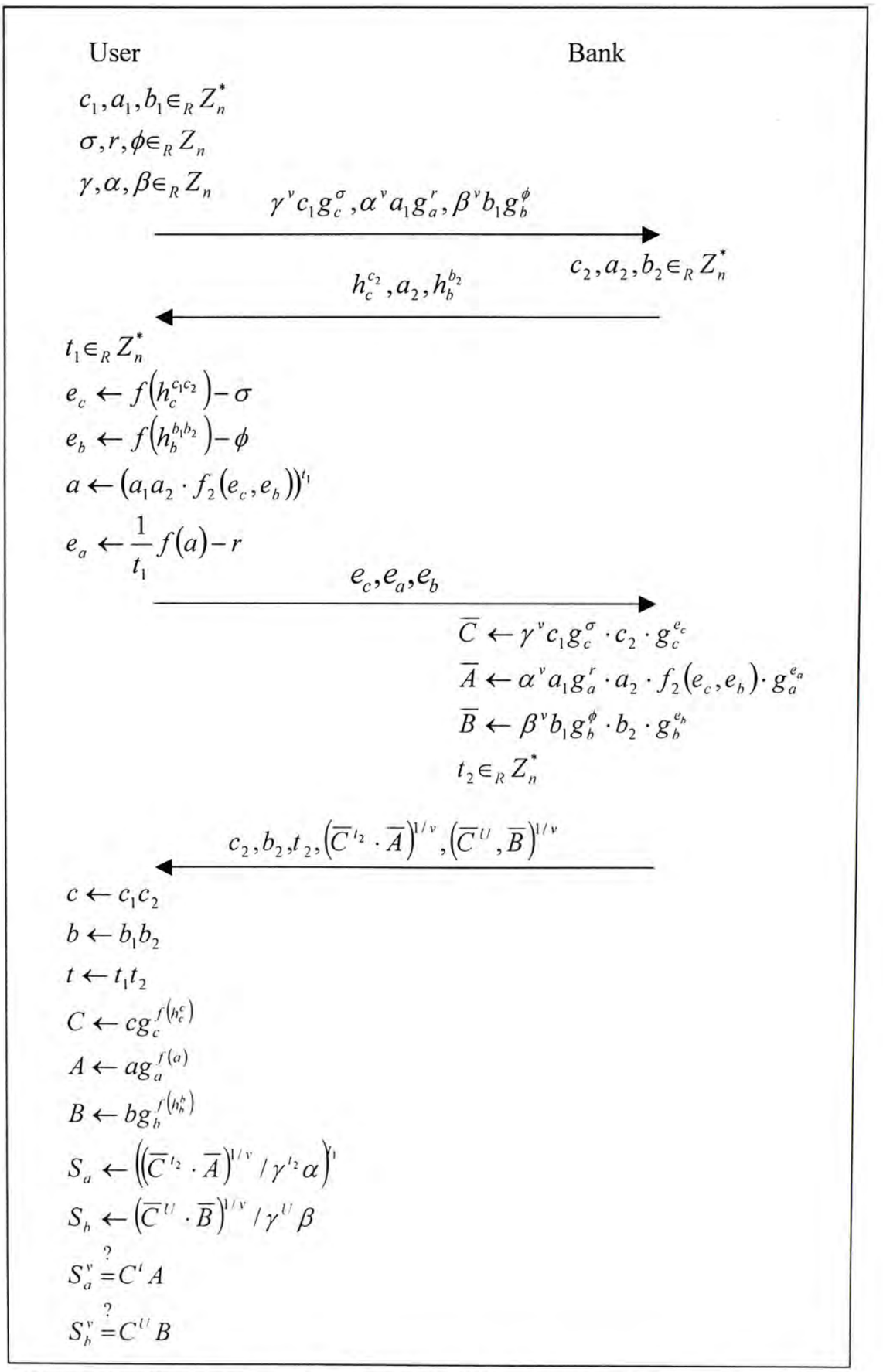

Figure 6.1: The withdrawal protocol of off-line transferable e-coin system

$e_c \leftarrow f\left(h_c^{c_1 c_2}\right) - \sigma$

$e_b \leftarrow f\left(h_b^{b_1 b_2}\right) - \phi$

$a \leftarrow \left(a_1 a_2 \cdot f_2(e_c, e_b)\right)^{t_1}$, where f_2 is an one - way function.

$e_a \leftarrow \frac{1}{t_1} f(a) - r.$

The user sends e_a, e_b and e_c to the bank.

Step 4 The bank computes three values $\overline{C}$, $\overline{A}$, $\overline{B}$ and sends them to the use, where

$\overline{C} \leftarrow \gamma^v c_1 g_c^{\sigma} \cdot c_2 \cdot g_c^{e_c}$, for $c = c_1 c_2$

$\overline{B} \leftarrow \beta^v b_1 g_b^{\phi} \cdot b_2 \cdot g_b^{e_b}$, for $b = b_1 b_2$

$\overline{A} \leftarrow \alpha^v a_1 g_a^{r} \cdot a_2 \cdot f_2(e_c, e_b) \cdot g_a^{e_a} = \alpha^v a g_a^{(1/t_1) f(a)}$

The bank sends $c_2, b_2, t_2, \left(\overline{C}^{t_2} \cdot \overline{A}\right)^{1/v}$, and $\left(\overline{C}^{U}, \overline{B}\right)^{1/v}$ to User, where t_2 is a random number.

Step 5 The user finds three numbers $C \leftarrow c g_c^{f\left(h_c^c\right)}$, $A \leftarrow a g_a^{f(a)}$, $B \leftarrow b g_b^{f\left(h_b^b\right)}$

And constructs two signatures

$S_a \leftarrow \left(\left(\overline{C}^{t_2} \cdot \overline{A}\right)^{1/v} / \gamma^{t_2} \alpha\right)^{t_1}$, $S_b \leftarrow \left(\overline{C}^{U} \cdot \overline{B}\right)^{1/v} / \gamma^{U} \beta$

She can verify the signatures by

$S_a^v \stackrel{?}{=} C^t A$ and $S_b^v \stackrel{?}{=} C^U B$ where $t = t_1 t_2$.

After withdrawal, User will obtain a coin with a value d. It include three base numbers (a, b, c) and two bank's signatures $(C^u B)^{1/v}$, $(C^t A)^{1/v}$.

6.3 The Transfer / Payment Protocol

Before the payment actually takes place. Payer and payee have to come to an agreement on what the object is going to be purchased and for which amount d.

A simplified payment protocol between User_k and User_{k+1} runs in figure 6.2. After the payment, a new coin with no values, which belongs to User_k, will append to the transferred coin. That means User_k's identity will be embedded into the coin.

Step 1 $User_k$ send $Coin_{k-1}$ $\{Co_{k-1}, LA_{k-1}, LB_{k-1}, LC_{k-1}, LR_{k-1}, d\}$ to $User_{k+1}$ where

$$LA_{k-1} = a_1 \| a_2 \| \ldots \| a_k$$
$$LB_{k-1} = b_1 \| b_2 \| \ldots \| b_k$$
$$LC_{k-1} = c_1 \| c_2 \| \ldots \| c_k$$
$$LR_{k-1} = r_1 \| r_2 \| \ldots \| r_{k-1}$$

Where (a_i, b_i, c_i) is the base numbers of the coin that embedded by $User_i$ in previous payments.

$User_{k+1}$ should verify the $coin_{k+1}$ by using function $\overset{?}{valid}(Coin_{k-1})$. The details of function is show as below:

Input:

$Coin_i : \{Co_i, LA_i, LB_i, LC_i, LR_i, d\}$

Temporary

$a_1, a_2, \ldots, a_{i+1}$	*seperate* LA_i
$b_1, b_2, \ldots, b_{i+1}$	*seperate* LB_i
$c_1, c_2, \ldots, c_{i+1}$	*seperate* LC_i
$r_1, r_2, \ldots, r_i$	*seperate* LR_i
$x_1, x_2, \ldots, x_i$	$x_j \leftarrow f_3(a_{j+1}, b_{j+1}, c_{j+1})$
$A_1, A_2, \ldots, A_{i+1}$	$A_j \leftarrow a_j g_a^{f(a_j)}$
$B_1, B_2, \ldots, B_{i+1}$	$B_j \leftarrow b_j g_b^{f\left(h_h^{b_j}\right)}$
$C_1, C_2, \ldots, C_{i+1}$	$C_j \leftarrow c_j g_c^{f\left(h_c^{c_j}\right)}$
Co_{temp1}	$Co_{temp1} \leftarrow C_1^{r_1} A_1^{x_1} B_1$
Co_{temp2}	$Co_{temp2} \leftarrow \prod_{j=2}^{i} \left(C_j^{r_j} A_j^{x_j} B_j\right)$

Output

$(Co_i)^{v_{d_0} \cdot v_d} \overset{?}{=} (Co_{temp1})^{v_{d_0}} (Co_{temp2})^{v_d}$

Where f_3 is an one-way and collision resistant hash function, v_{d_0} and v_d

are public exponents associate with zero amount and d amount repectively. Please note that, for first payment, $User_1$ pays $Coin_0$ to $User_2$, where $Co_0=1$, $LA_0=a_1, LB_0=b_1, LC_0=c_1, LR_0=0$ and d is the denomination of coin that $User_1$ pays to $User_2$.

Step 2 $User_{k+1}$ sends a challenge x_k to $User_k$ where $x_k \leftarrow f_3(a_{k+1}, b_{k+1}, c_{k+1})$

In this step, the challge x_k is an output of an one-way hash function f instead of a random generated number. The method has two purposes

1. Ensure $User_k$ always get different challenges if she tries to spend a coin more than once.
2. Ensure $User_{k+1}$ only can use one secret value in chalenge and response method of later payment.

Step 3 $User_k$ computes the response $r_k \leftarrow t_k x_k + U_k$, a zero value coin $Co'_k \leftarrow (S_{ak})^{x_k} S_{bk}$, and sends them to $User_{k+1}$.

Step 4 $User_{k+1}$ verifies the response by finding wether $(Co'_k)^{v_{d_0}}$ equal to $C_k^{r_k} A_k^{x_k} B_k$, if it is not equal, $User_{k+1}$ terminates the protocol.

Until now, $User_{k+1}$ can produce a valid new coin $Coin_k$ with amount d. She does following job:

$$LA_k = LA_{k-1} \| a_{k+1}$$
$$LB_k = LB_{k-1} \| b_{k+1}$$
$$LC_k = LC_{k-1} \| c_{k+1}$$
$$LR_k = LR_{k-1} \| r_k$$
$$Co_k \leftarrow Co_{k-1} \cdot Co'_k$$

$User_{k+1}$ obtains $Coin_k$ with form $\{Co_k, LA_k, LB_k, LC_k, LR_k, d\}$.

If we want to pay several coins to match a total amout D, we can run payment protocols parellelly to save the processing time.

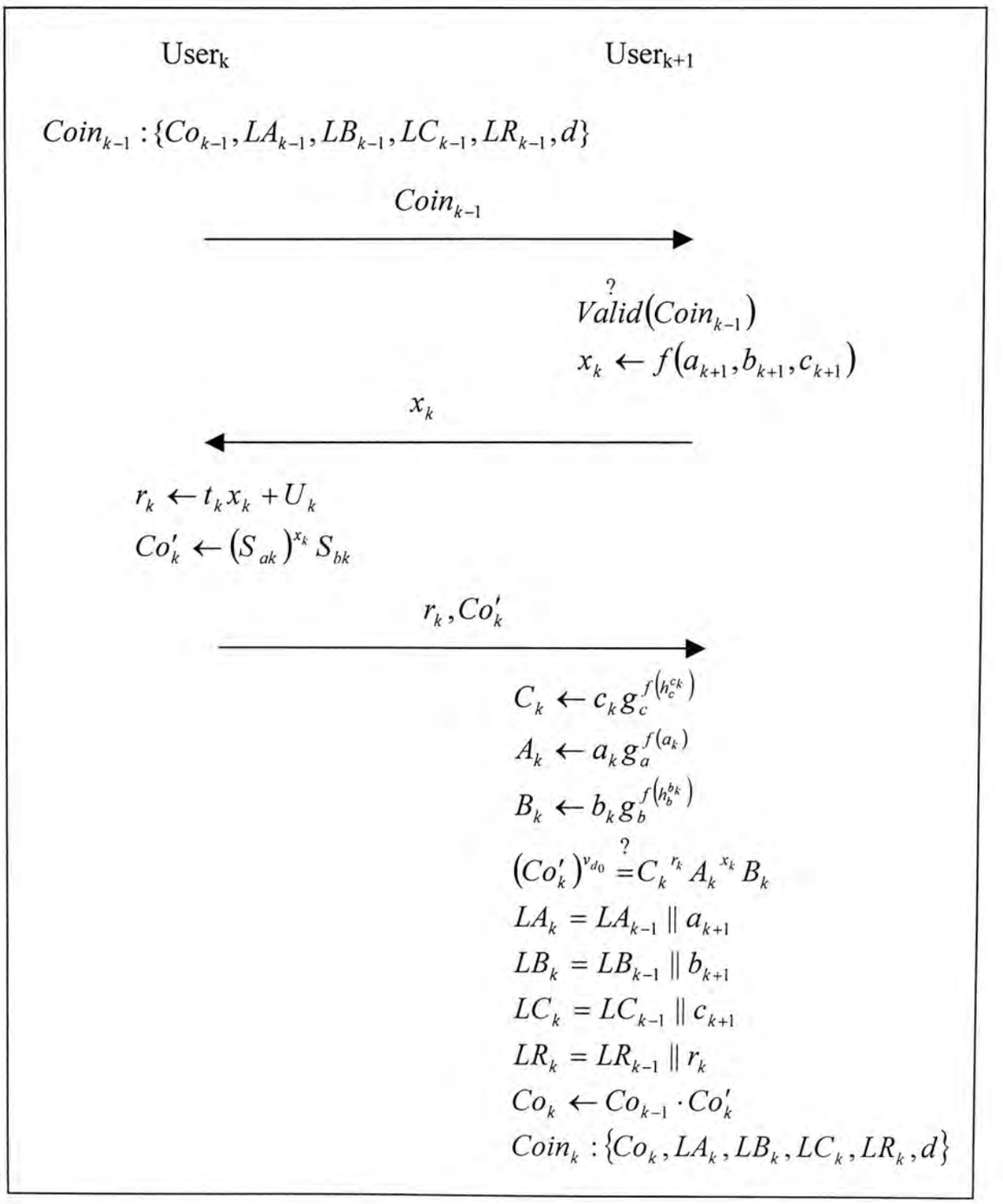

Figure 6.2: The Payment Protocol between User$_k$ and User$_{k+1}$

6.4 The Deposit Protocol

After k times of transfer, $user_k$ want to deposit the coin into the bank, she first connects to the bank and sends the $coin_{k-1}$. The bank checks wether the coin is valid by using the function $\overset{?}{valid}(Coin_{k-1})$ (function details please refer to section 6.3). If the coin is valid, the bank deposits the money of amount d into $User_k$'s account. In the deposit protocol (figure 6.3), we don't need to keep user anonymous, as $User_k$ should present his account number to the bank. In order to offer unconditional privacy, $User_k$ cannot use $coin_k$, which was used to generate challenge x_{k-1} for last payment, any more. Otherwise, the bank can link the payment to $User_k$'s account as the coin generates same challenge.

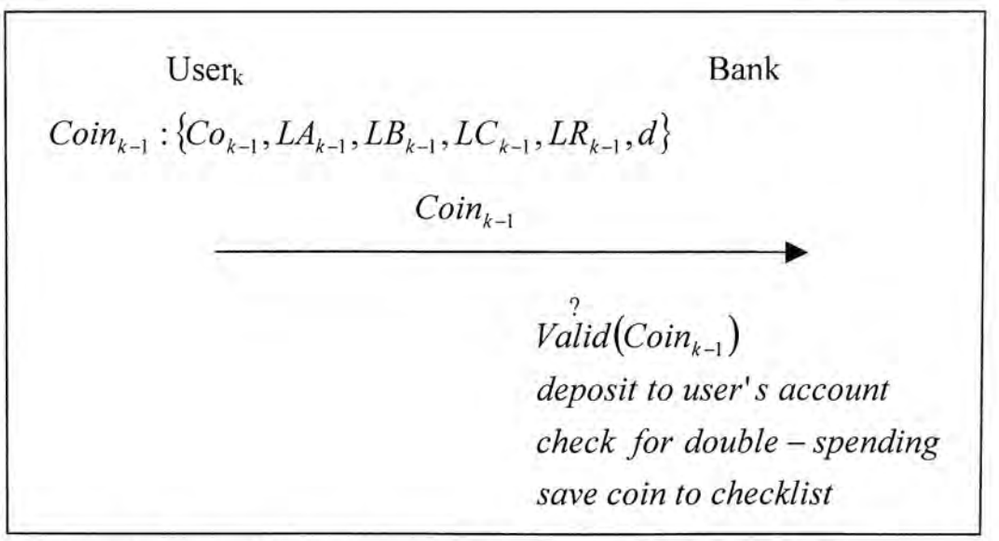

Figure 6.3: The Deposit protocol for between user and bank

The most important part in deposit protocol is how to find the double-spender. Consider the tree of payments constructed in Figure 6.4. $User_1$ withdraws a $Coin_0$ from the bank and pays this coin to $User_2$. During this transaction $Coin_0$ is changed to $Coin_1$. Later $User_2$ pays this coin to $User_3$, and after this transaction coin has been changed to $Coin_2$. $User_3$ double spend the money and pays the coin to $User_4$ and $User_4$', after transactions, coin became $Coin_3$ and $Coin_3$'. $User_4$ and $User_4$' continue pays the coin to other users and so on. Later, $User_6$', $User_7$" and $User_6$ deposit the coins $Coin_5$', $Coin_6$"

and $Coin_5$. From the payment tree, you can see that both $User_3$ and $User_5$ double-spend the money. By using the transfer protocol described previously, we can find that the bank only can find the identities of double-spender and other users' identity is unknown.

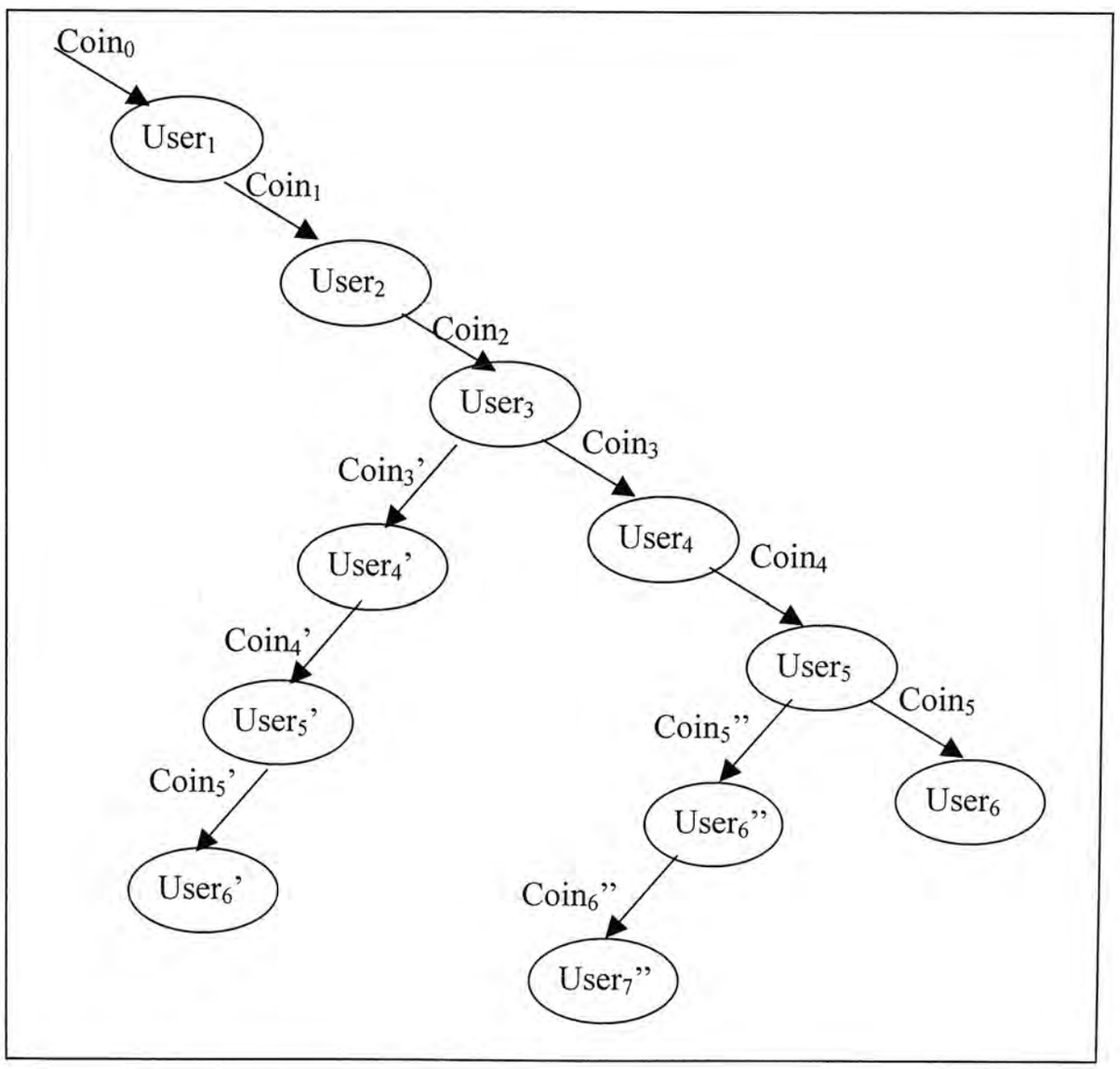

Figure 6.4: Payment tree with double-spending

The bank will get a list of coin's base numbers (*a, b, c*), challenges *x* and responses *r* from each coin.

From $Coin_5$', the bank gets:

$(a_1, b_1, c_1, r_1, x_1), (a_2, b_2, c_2, r_2, x_2), (a_3, b_3, c_3, r_3', x_3'), (a_4', b_4', c_4', r_4', x_4'), (a_5', b_5', c_5', r_5', x_5'),$
(a_6', b_6', c_6', U_6')

From $Coin_6$", the bank gets:

$(a_1, b_1, c_1, r_1, x_1), (a_2, b_2, c_2, r_2, x_2), (a_3, b_3, c_3, r_3, x_3), (a_4, b_4, c_4, r_4, x_4), (a_5, b_5, c_5, r_5'', x_5''),$
$(a_6'', b_6'', c_6'', r_6'', x_6''), (a_7'', b_7'', c_7'', U_7'')$

From $Coin_5$, the bank gets:

$(a_1, b_1, c_1, r_1, x_1), (a_2, b_2, c_2, r_2, x_2), (a_3, b_3, c_3, r_3, x_3), (a_4, b_4, c_4, r_4, x_4), (a_5, b_5, c_5, r_5, x_5),$
$(a_6, b_6, c_6, r_6, x_6)$

Bank will search each coin base numbers in the checklist, if coin's owner has double-spent the coin, the same coin base number will have a pair of different challenge and response, in this case, they are coin with base number (a_3, b_3, c_3) and (a_5, b_5, c_5). The identity of users can be found easily from challenge and response pairs using Polynomial Secret Sharing Scheme (detail please refer to section 4.1.2).

After checking for double-spender, the bank saves the coin list to its checklist for later comparison of the double-spender checking.

6.5 Expansion of Coins

Until now, we are talk about transfer the coin with fixed value. In each transaction, every user adds a coin with no value into original coin. In this chapter, we will introduce a new concept that user can expand the denomination of coin during the transaction. The mechanism we are talking about here is really a very simple trick: Instead of adding a coin with no value, user can add a coin with desired value. We just need to make a little change on the original protocol. The form of $Coin_k$ will become $\{Co_k, LA_k, LB_k, LC_k, LR_k, LD_k\}$ where $LD_k = \hat{d}_1 \| \hat{d}_2 \| \ldots \| \hat{d}_k$ ($\hat{d}_i$ is the denomination of each attached $Coin_i$). As our denomination distribution table in chapter 6.3, there are only 10 different denominations. The size of LD_k will be very small compare with RSA signature, hence the size of whole coin will not change a lot.

Also the verify function $\overset{?}{valid}(Coin_i)$ (detail please refer to chapter 6.4) will change a little too.

Input:

$$Coin_i : \{Co_i, LA_i, LB_i, LC_i, LR_i, LD_i\}$$

Temporary:

$a_1, a_2, \ldots, a_{i+1}$	*seperate* LA_i
$b_1, b_2, \ldots, b_{i+1}$	*seperate* LB_i
$c_1, c_2, \ldots, c_{i+1}$	*seperate* LC_i
$r_1, r_2, \ldots, r_i$	*seperate* LR_i
$\hat{d}_1, \hat{d}_2, \ldots \hat{d}_i$	*seperate* LD_i
$x_1, x_2, \ldots, x_i$	$x_j \leftarrow f(a_{j+1}, b_{j+1}, c_{j+1})$
$A_1, A_2, \ldots, A_{i+1}$	$A_j \leftarrow a_j g_a^{f(a_j)}$
$B_1, B_2, \ldots, B_{i+1}$	$B_j \leftarrow b_j g_b^{f\left(h_h^{h_j}\right)}$
$C_1, C_2, \ldots, C_{i+1}$	$C_j \leftarrow c_j g_c^{f\left(h_c^{c_j}\right)}$
$Co'_{d_0}, Co'_{d_1}, \ldots Co'_{d_9}$	$Co'_{d_l} \leftarrow \prod_{j=1}^{i} Co_{tempj}$
	$\begin{pmatrix} where\ Co_{tempj} \leftarrow C_j^{r_j} A_j^{x_j} B_j\ if\ \hat{d}_j = d_l, \\ else\ Co_{tempj} = 1 \end{pmatrix}$

Output

$$(Co_i)^{\prod_{j=0}^{9} v_{d_j}} \overset{?}{=} \prod_{j=0}^{9} \left(Co'_{d_j}\right)^{\prod_{l=0, l\neq j}^{9} v_{d_l}}$$

Compared with fixed value coin transaction, transfer coin with expansion need more exponentiation computation. The deposit protocol between the user and the bank will remain unchanged.

6.6 Security and privacy Analysis

In this section we analyze the issues on the security and privacy of the payment

system.

Privacy

The privacy of payment may be the decisive factor in ultimate acceptance of electronic payment system in an open market. In the transferable e-coin system protocol proposed, we offer unconditional privacy. The payment is **confidential**, the wire-tapper cannot get any details of transaction, and even the bank cannot get any information such as purchase descriptions. The payment have **untraceability**, every electronic cash payment doesn't cause a unique transaction identifier to end up in the computer files of the bank.

In particular, privacy is extremely difficult in implementation of a transferable e-cash system, because cash embed a list of users' identity, in order to offer the privacy, system should only allow the double-spenders can be identified and other users' identity are hidden from the bank. Our payment system satisfies this condition, it achieves the property of privacy.

Change of Coin Content:

A possible attack is double-spender will modify the coin content such that bank cannot find her identity. Let's consider Coin_k with the form $\{Co_k, LA_k, LB_k, LC_k, LD_k, LR_k\}$ (we assume here user can expand coin during transfer, if the user cannot crack such coin, it is obviously that she cannot crack the coin which transferred in fixed value). Can user make a new Coin_k' without adding any new coin and have the form $\{Co'_k, LA'_k, LB'_k, LC'_k, LD'_k, LR'_k\}$, which can pass the verify function $\overset{?}{valid}(Coin_k)$? Co_k is a multiplication of a list of valid Randomized Blind Signature, As the property of the Signature, the only way the user can change the value of Co_k is multiply it with a valid coin. By doing this, user's identity will be embedded into the coin. Hence the user cannot generate a new Co'_k without involve his identity. How about change the values of LA'_k, LB'_k, LC'_k ? Ferguson's paper [Fer93] shows that the difficulty of

changing value of *a, b, c* is same as forge a randomized blind signature. LD'_k is the public key of the bank for different denomination is co-prime to each other. Forging LD'_k is stated to be equivalent in computational difficulty to factoring. Now, let's think about LR'_k, it represents the response assume came from user's secret value and her ID. If the user can change the value of LR'_k, the bank cannot find it's identity even user double spend the coin. Let's look at the value of Co'_k, it equal to:

$$\left(C_1^{r_1} A_1^{x_1} B_1\right)^{1/v_{d_1}} \cdot \left(C_2^{r_2} A_2^{x_2} B_2\right)^{1/v_{d_2}} \cdots\cdot \left(C_k^{r_k} A_k^{x_k} B_k\right)^{1/v_{d_k}}$$

As we proof in chapter 5, user can not forge the content of the each signature with co-prime public exponent even several signatures are multiply together. Hence we can divide Co'_k into terms depend on their exponents. Each terms will look like below:

$$\left(C_1^{r_1} A_1^{x_1} B_1 C_2^{r_2} A_2^{x_2} B_2 \cdots C_i^{r_i} A_i^{x_i} B_i\right)^{1/v_d}$$

Since user cannot change the value of *A, B, C* and *x*. The only way he can change LR'_k is to find a list $r'_1, r'_2, \ldots r'_i$ such that $C_1^{r'_1} \cdot C_2^{r'_2} \cdots C_i^{r'_i} = C_1^{r_1} \cdot C_2^{r_2} \cdots C_i^{r_i}$. One way to change the exponent is to find a set of roots that is multiplied together equal to the product of other set of roots. The probability of find such two sets successfully is 2^{-T} where T is proportional to the time and space required. Otherwise, As far as we know, this problem is equivalent in computational difficulty to factoring (the problem is detailed mentioned in [CEG87]).

Coin Double-spending

As we know, every user can has not only one but a number of coins. That means User will have a list of secret values that embed in each coin. Let's think about bellow situation:

A user want to double spend the money, she embed one coin in first payment, and embed the other coin in second payment. When two coins are deposited, the bank will find the coin is double-spent, but can not find the double-spender, since it will be 3 unknown variables in 2 polynomials.

Let's look at our protocol to see if it can really happen. Before the user receive a coin, she should send out a challenge x, which is an output of a one-way function f_3 with input a, b, c where a, b, c is the base numbers of the coin that should be used next time. This method restricts the coin that will be used in next payment. Hence. The above situation will not happen.

Generation of *a, b, c*

If User can find two coins *Coin* and *Coin'* such that their base numbers satisfy $f_3(a,b,c) = f_3(a',b',c')$, double-spending will become very easy. Since f_3 is an one-way and collision resistant function (detail please refer to section 2.6), it is impossible to find the other triple (a', b', c') that has the same output as triple (a, b, c). So can User find two coins such that their base number is the same? The answer is negative. The withdrawal protocol makes sure that base numbers is randomly chose, both bank and user can not control the values of (a, b, c). The probability of a user gets two coins with same base numbers is very small, approach to zero.

Framing by the Bank

In order to protect user against framing by the bank we need a little modification on the protocol. Let U be the concatenation of user's identity and a unique coin number. This makes the U's of all the coins distinct. If the bank want to accuses a user of having doubles-spent a coin, the bank has to tell the challenges and responses (say 2 pairs of (x, r)) that found with corresponding a, b and c (all of them are verifiable by a third party.). If the user didn't spend the coin with that identity twice, bank have no knowledge of a, b, or c. So if the bank tries to frame the user, the triple (a, b, c) will be different (with high probability) from the actual values used by the user. If the user can provide a coin with different triple (a, b, c) that match the value of U, then the bank must be framing the user, as she cannot generate a new triple which matches the same value of U.

Late Detection of double-spender

In single payment, double spender can be detected soon because the coin only spent once. But for transferable e-coin, it has different case. Each transfer delays the detection of double-spender. Double-spender will not be noticed until two copies of the same coin are deposited and it may be too late by then. We can solve this problem by adding timestamps (details please refer to section 2.9) into each valued coin, the time-stamping will indicate the time that the coin is withdrawal. When user finds the coin is too old to use, he should deposit it into the bank.

Coin Recognition

Users can recognize their coin if they see it later in another payment. It is acceptable, since in the real world, we also have the ability to recognize the paper cash when we receive it again.

Unfortunately, we cannot prove that there is no other way for user to cheat. The attacks described above are only the most obvious ones. At present, the state of the art in cryptography does not allow us in general to prove the security of such a protocol.

6.7 Complexity Analysis

In this chapter we analyze the complexity of the proposed transferable e-coin system.

Coin grows in size

Any transferable electronic cash system has the property that the coin must grow in size [ChaPe93] each time it is spent because of the information it has to contain. This information is about every person who has spent the coin for the bank to maintain its ability to catch multiple spenders. This limits the maximum number of transfers allowed in the system by the allowable size of the coin. In the protocol, we assume that the Bank public a number l that is the maximum number of transfers. If a user receives

a coin, which was transferred for l times, it means the size of coin is too large to use, the user should deposit the coin and cannot pay it anymore. Other user has the right to refuse this coin because its size is not suitable for further payments.

Divisibility of the Coin

The system described so far in this report does not have the ability to split the coin into smaller sub parts. The main reason of we don't add divisibility into system is that the whole system will become very complex, which makes system extremely difficult to understand, verify, implement or debug.

Checklist of Bank

To detect double-spending, the bank should store all base numbers, challenges and responses of deposited coins in the checklist. The bank needs a huge storage database, and the size of database grows day by day. To solve this problem, we add a time limit to coins, for example one year, the coin should be deposit before the time limit. After the time limit, the coin will become invalid and cannot be use anymore. This method can prevent bank's database from growing unlimitedly. Moreover, This method saves the computational power and increases system's security.

Computational Efficiency

As we can see, the user should verify all previous payments during the transaction. For one more transfer, the user need to multiply two more terms which has the power x and r respectively, i.e. the user needs to compute out the value of $\prod_{i=1}^{k} g_i^{a_i}$ (where k is increase by 2 after one transfer) in each transaction. Using vector addition chain techniques (see [Cos90] and [Oli81]), we can efficiently compute the value of $\prod_{i=1}^{k} g_i^{a_i}$ in RSA system even for large k. The basic steps are as follow:

Step 1 Input $(g_1, g_2, \ldots g_k)$ and $(a_1, a_2, \ldots, a_k)$

Step 2 Sort the k elements $(g_1,a_1),\ldots,(g_k, a_k)$ in descending order on the a_i's. (The new sequence $(g_1,a_1),\ldots,(g_k, a_k)$ is such that $a_1 \geq a_2 \geq \cdots \geq a_k$).

Step 3 For i=1 to k-1 do $a_i \leftarrow a_i - a_{i+1}$, $g_{i+1} \leftarrow g_i g_{i+1}$. If "stop-condition" holds then output $\prod_{i=1}^{k} g_i^{a_i}$ else go Step 2.

Please note that the stop-condition depends on how small the index-tuple has become. The meaning of small itself depends on whether one can apply table lookup or do precomputations and so on. In general, it depends on the particular environment the algorithm is implemented in.

Size of Base Numbers

In our system, one coin have three base numbers a, b, and c, the storage size will be smaller if we can generate C, A, and B under one-way functions of a single base number c. Then we could store a single coin in about 70 bytes [Fer93]. However, constructing an efficient withdrawal protocol for such a coin remains an open problem.

6.8 Conclusion

In this chapter we introduced a new off-line transferable e-coin system motivated by Ferguson's single-term off-line coins system. Although the ability to transfer paper cash is very important in our daily life, there are only a little electronic payment papers talking about transferability because of its difficulty in implementation. The proposal of single-term e-cash protocol makes application of transferable on e-cash system become possible. In our protocol, Anonymous is protected by using Randomized Blind Signature and double-spending is prevented by using polynomial secret sharing scheme. The protocol we proposed here is the first single-term transferable e-cash, which is more efficient and simpler than the old multiple-term e-cash system.

Chapter 7
A New Off-line E-check System

7.1 Introduction

Although the basic Ferguson's payment protocol is quite efficient and does not require much storage space if the user pays at a shop with a bunch of coins, it is considerable quite expensive (in complexity) if we want to pay the money with amounts that can only be paid with many coins are involved. For this purpose, we propose an electronic check system in this chapter.

In the basic coin system, the user withdraws a fix amount of money from the bank and cannot change the value of each coin during the transaction. The basic unit in the off-line payment system that we going to create is called check, which can be used for a variable amount, not determined at the withdrawal transaction. The way we implement the property of e-check, essentially, is that we make the user pay the maximum value of the check at the withdrawal time, and give him a refund for the part of the check that he didn't use at the payment transaction.

Until now, several e-check schemes including on-line systems [Cha90] and off-line systems [VA90, ChaDB90, Bra93] are proposed. Most of them are using cut-and-choose methodology except Brands' scheme [Bra93]. The scheme we present here is more efficiency than the Brands' scheme.

In this chapter, we present two e-check systems: one is very simple and efficient but only support partial anonymous; the other one is more complex and supports both

privacy and untraceability.

7.2 E-checks Models

The transaction model of an e-check system is slight different compare with the one we introduced previously (see Figure 7.1). Here we find that there is an extra transaction, called refund transaction.

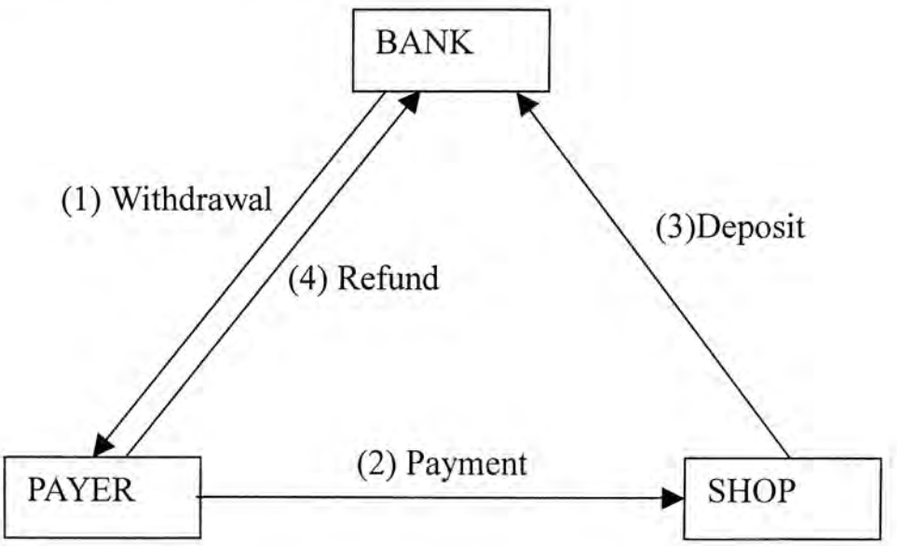

Figure 7.1: The transaction model of e-check system

The user first gets a check from the bank by the execution of a **withdrawal** protocol. The amount of check is decided by the user, which present the maximum the amount of money that the user can pay to the shop. The bank charges the user, by taking the equivalent amount of "traditional" money out of user's bank account. After the withdrawal, the check is transferred from the user to the shop in a **payment** transaction. Before the check is actually given to the shop, the user will adjust the value of the check to the amount agreed by both the shop and the user. (Please note that, the payee's representation of electronic check need not necessarily be the same as that for the payer, and in fact it preferably is not.) The shop then **deposits** the electronic

check from the bank with the equivalent amount of money in the payment. Since the user only uses a part of the check in the payment transaction, she can **refund** her money from the bank with the value she didn't use.

The refund transaction of the last batch of checks is usually done together with the withdrawal transaction of the next batch. There is no transferability in this model. More about this will be discussed at the end of this chapter.

7.3 E-Check System with Partial Privacy

In this chapter, we introduce a new e-check system, which is much simpler than previously proposed e-check system. Paying the check with a desired amount is just as efficient as a single coin. But the system only supports partial anonymous and does not satisfy perfect untraceability, i.e. the bank can link the payment and the user together when the user want to refund the check. But the system is still useful in some applications that untraceability does not provide any problems.

7.3.1 The Withdrawal Protocol

The withdrawal protocol is basically same as the Ferguson's withdrawal protocol with small modification on the denomination scheme. In chapter 5.1, we defined a list of distinct public exponents associated with a list of denominations. In the e-check system we also have a list of co-prime numbers ($v_1, v_2, \ldots, v_k$), with v_i^{th} root representing some amount of money, say $\$2^{i-1}$. So multiplying the appropriate roots according to the binary expansion of the particular amount can represent any amount up to $\$2^k$-1. For example, a 5^{th} root worth \$2, a 7^{th} root worth \$4, and a 35^{th} root worth \$6.

Before the withdrawal (figure 7.2) actually takes place. The user and the bank have to come to an agreement on which amount of money that will be withdrawn and the bank uses the corresponding key to make the signature. In the protocol described below, we assume that the user wants to withdraw an e-check with amount $\$2^k$-1, and

the signature can be modified by public key $v_1 \cdot v_2 \cdots\cdots v_k$.

Step 1 The user chooses randomly a_1, c_1, b_1 (her three contributions to the base numbers) $\sigma, r, \phi, \alpha, \beta$ *and* γ (the blinding factors). She computes $\gamma^{v_1 v_2 \cdots v_k} c_1 g_c^{\alpha}$, $\alpha^{v_1 v_2 \cdots v_k} a_1 g_a^{r}$, $\beta^{v_1 v_2 \cdots v_k} b_1 g_b^{\phi}$ and sends them to the bank.

Step 2 The bank then chooses her three contributions to the base numbers c_2, a_2, b_2. And sends $h_c^{c_2}, h_b^{b_2}$ and a_2 to the user. Sending a_2 directly allows the user to raise one of the resulting signatures to a power she chooses.

Step 3 The user chooses a random number t_1 and computes

$e_c \leftarrow f\left(h_c^{c_1 c_2}\right) - \sigma$

$e_b \leftarrow f\left(h_b^{b_1 b_2}\right) - \phi$

$a \leftarrow \left(a_1 a_2 \cdot f_2(e_c, e_b)\right)^{t_1}$, where f_2 is an one - way function.

$e_a \leftarrow \frac{1}{t_1} f(a) - r.$

The user sends e_a, e_b and e_c to the bank.

Step 4 The bank computes the blinded version of A, B and C.

$\overline{C} \leftarrow \gamma^{v_1 v_2 \cdots v_k} c_1 g_c^{\sigma} \cdot c_2 \cdot g_c^{e_c}$, for $c = c_1 c_2$

$\overline{B} \leftarrow \beta^{v_1 v_2 \cdots v_k} b_1 g_b^{\phi} \cdot b_2 \cdot g_b^{e_b}$, for $b = b_1 b_2$

$\overline{A} \leftarrow \alpha^{v_1 v_2 \cdots v_k} a_1 g_a^{r} \cdot a_2 \cdot f_2(e_c, e_b) \cdot g_a^{e_a} = \alpha^{v_1 v_2 \cdots v_k} a g_a^{(1/t_1) f(a)}$

The bank then chooses a random number t_2 and sends $c_2, b_2, t_2, \left(\overline{C}^{t_2} \cdot \overline{A}\right)^{1/v_1 v_2 \cdots v_k}$, and $\left(\overline{C}^{U}, \overline{B}\right)^{1/v_1 v_2 \cdots v_k}$ to User.

Step 5 The user now constructs the numbers

$C \leftarrow c g_c^{f\left(h_c^c\right)}$, $A \leftarrow a g_a^{f(a)}$, $B \leftarrow b g_b^{f\left(h_b^b\right)}$

and computes two signatures

$S_a \leftarrow \left(\left(\overline{C}^{t_2} \cdot \overline{A}\right)^{1/v_1 v_2 \cdots v_k} / \gamma^{t_2} \alpha\right)^{t_1}$, $S_b \leftarrow \left(\overline{C}^{U} \cdot \overline{B}\right)^{1/v_1 v_2 \cdots v_k} / \gamma^{U} \beta$

She checks the signatures she received are correct by verifying that

$S_a^{v_1 v_2 \cdots v_k} \stackrel{?}{=} C^{t} A$ and $S_b^{v_1 v_2 \cdots v_k} \stackrel{?}{=} C^{U} B$ where $t = t_1 t_2$.

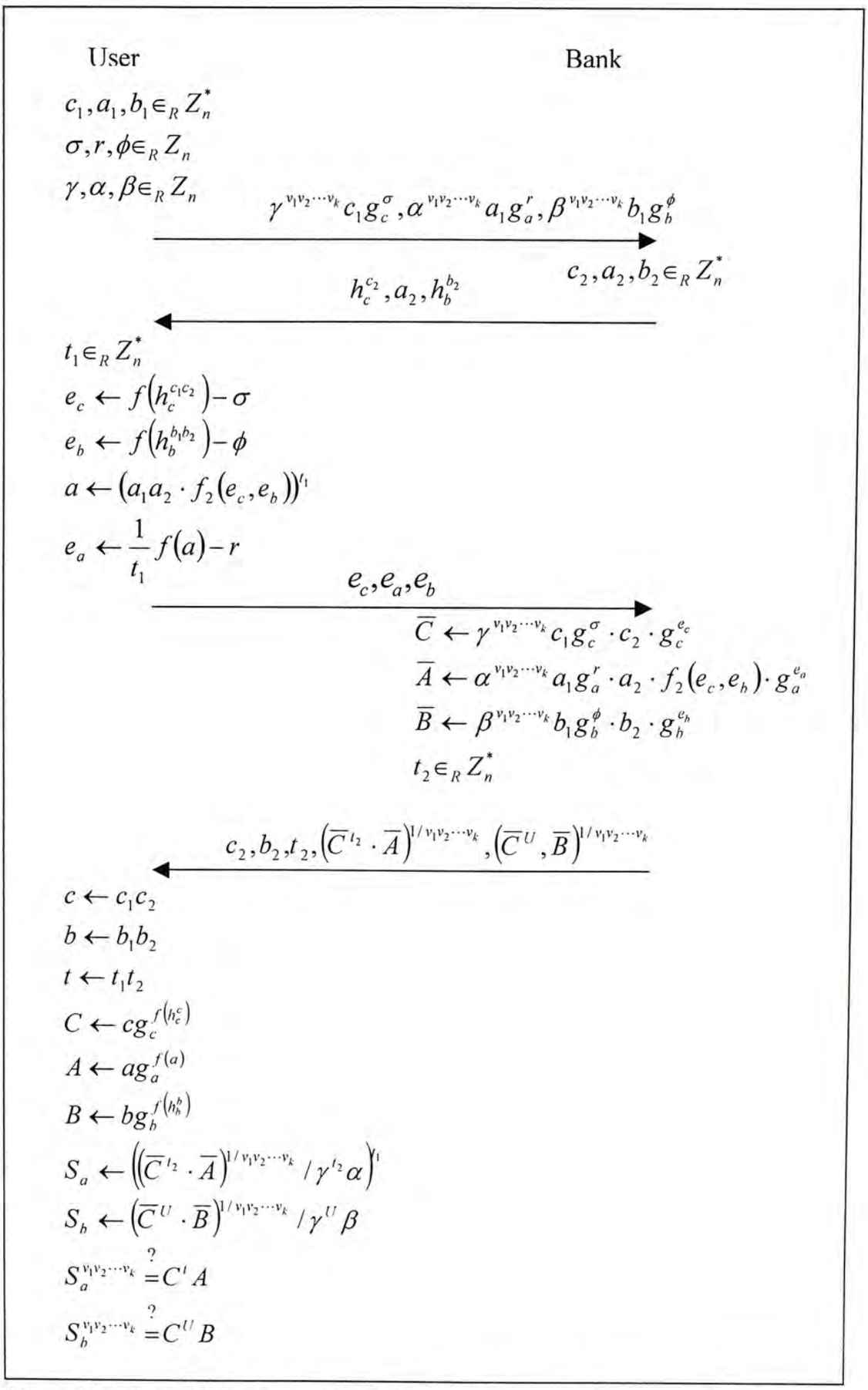

Figure 7.2: The withdrawal protocol of e-check system with partial privacy

The user ends up with the following set of numbers: a, b, c, t, S_a and S_b which are the 3 base numbers, the random parameter for the secret sharing line and the 2 signatures.

7.3.2 The Payment Protocol

Before the execution of actual payment, the user needs to devalue the check to the desired amount. To maintain protocol's generality in figure 7.3, we assume that the user wants to pay $\$2^i$-1 to the shop, where $1 \le i \le k$.

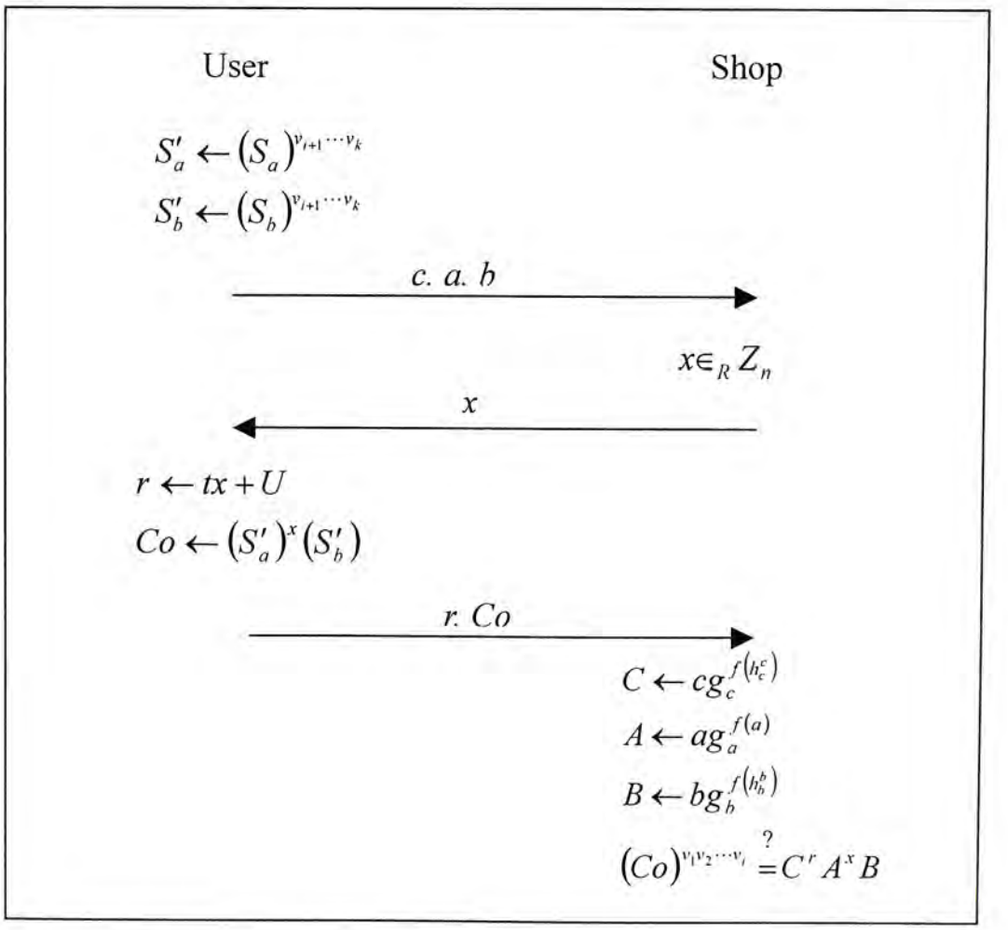

Figure 7.3: The payment protocol of e-check system with partial privacy

Step 1 The user devalues the check from $\$2^k$-1 to $\$2^i$-1 by compute out

$S'_a \leftarrow (S_a)^{v_{i+1}\cdots v_k}$

$S'_b \leftarrow (S_b)^{v_{i+1}\cdots v_k}$

Step 2 The user sends *a, b*, and *c* to the shop.

Step 3 The shop replies with a randomly chosen challenge *x*.

Step 4 The user computes $r \leftarrow tx + U$ and $Co \leftarrow (S'_a)^x (S'_b)$.

And sends the results to the shop.

Step 5 The shop verifies the consistency of these two responses by

$C \leftarrow cg_c^{f(h_c^c)}$

$A \leftarrow ag_a^{f(a)}$

$B \leftarrow bg_b^{f(h_b^b)}$

$(Co)^{v_1 v_2 \cdots v_i} \stackrel{?}{=} C^r A^x B$

and then she accepts the payment.

After the payment, the shop saves 3 base numbers, the challenge, the response, the bank's signature and the amount of the check for later deposit.

7.3.3 The Deposit Protocol

The deposit protocol, which is shown in Figure 7.4, is very simple. It only needs one communication.

Step 1 The shop sends *c, a, b*, the challenge *x*, the responses *r*, the signature *Co* and the amount of check, i.e. $\$2^i$-1, to the bank.

Step 2 The bank verifies the correctness of the check and credits shop's account with corresponding amount.

Step 3 The bank searches through the checklist to find if the check with base numbers (*a, b, c*) have been double-spent before. The bank can easy find the double-spender, since she must reveal two different points on the line

$r \leftarrow tx + U$ which immediately allows the bank to determine her identity U.

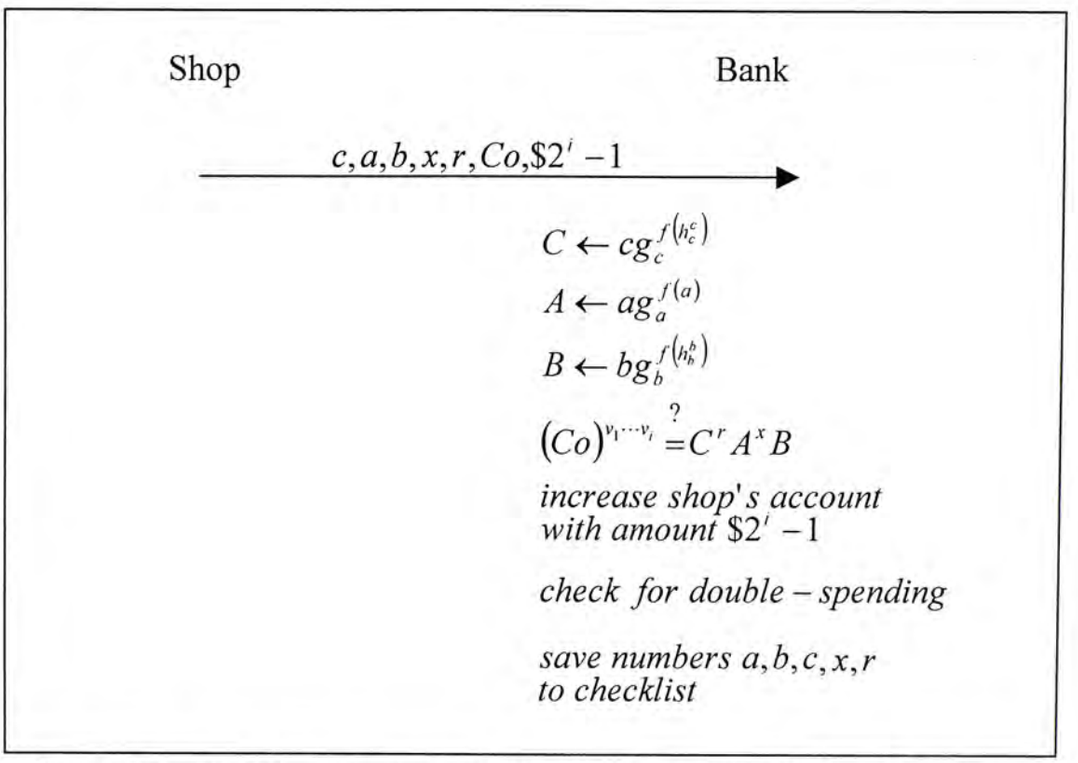

Figure 7.4: The Deposit protocol of e-check system with partial privacy

7.3.4 The Refund Protocol

When the user wants a refund for the amount of the check which she did not spent. She first informs the bank her account number and the amount that she wants to refund (in this case, it is $\$2^k$-1-$2^i$-1).

Step 1 The user sends check's base numbers (a, b, c), r, x, and $(C^r A^x B)^{1/v}$ to the bank.

Step 2 The Bank verifies the check similar the way that coins are verified (see section 4.4.2).

Step 3 The bank search the check in the list and verifies that the amount of the check with base numbers (a, b, c) have been spent less than or equal to $\$2^i$-1. Than the bank refunds the money to user's account.

Step 4 The bank stores the check with user's account number to prevent later use.

Notice the user can refund multiple checks at the same time. And the refund transaction is usually done together with the withdrawal transaction of the next checks.

7.3.5 Protocol Discussion

System Security

In the protocol, we fund that the user can obtain a list of Bank's signatures. Can the user obtain the other signature $x^{1/vi}$ that represents a larger amount of money? The answer is negative. A result of Shamir [Sha83] states that it is feasible for A to compute $x^{1/m}$ from $(x, x^{1/a1},\ldots, x^{1/ak})$ if and only if *m* divides the least common multiple of $(a_1,\ldots,a_k)$.

Partial Privacy

This protocol only supports partial anonymous, the shop doesn't know the identity of the user, but the bank can trace the check after it is refunded. She should show the value of (a, b, c), in case that the bank can verify the check. Since the base numbers are unique for each check (as we discussed in chapter 6, the probability of two checks have the same base numbers approach to zero), the bank can search the checklist to find which shop deposit the check with same base numbers. Once the bank finds the deposited check, the bank can link user's identity and her payment together.

Computational Efficiency

The withdrawal and payment protocol is as efficient as paying a single coin. The extra method that the user needs to do is devaluing the check to a suitable amount before payment. And refund protocol is very simple too; it only needs one communication between the bank and the user.

7.4 E-Check System with Unconditional Privacy

In this chapter we will show how to incorporate anonymity to our e-check system that means even the bank cannot trace the relation between the user and the payment. Privacy is the most difficult part in designing a E-check system. If user want to refund the check, how does she prove the valid of the check without showing the unique numbers which allow the bank to have the ability to trace the payment? Hence in our new e-check protocol, the actual check system is somewhat more complicated than the previous one.

In this new protocol, we use the other way to represent the denomination of the checks. We use a generator-turple $(g_{a_1}, g_{a_2}, \ldots, g_{a_k})$ of length k, representing some amount of money, say $\$2^{i-1}$. So any amount up to $\$2^k$-1 can then be represented with different combination of generators.

The check with maximum amount $\$2^k$-1 has k+1 signatures. One presents user's identity and remains are denomination parts each term presents an amount of $\$2^{i-1}$.

$$\text{The identity part :} \quad (C^U B)^{1/v},$$

$$\text{The denomination parts :} \quad (C^{t_1} A_1)^{1/v}, \ldots, (C^{t_k} A_k)^{1/v}$$

where t_i is secret value used in random challenge and response process. The numbers C, B, A_i are of the form $cg_c^{f(h_c^r)}$, $bg_b^{f(h_b^b)}$, $a_i g_{a_i}^{f(a_i)}$ respectively where the $f()$ is a suitable one-way function and all g's are publicly known elements or large order in the RSA group. We will show the protocol details in coming sections.

7.4.1 The Withdrawal Protocol

In this section, we will show how the withdrawal protocol from section 4.2.1 can be modified in a fairly straightforward manner for an e-check system (figure 7.5).

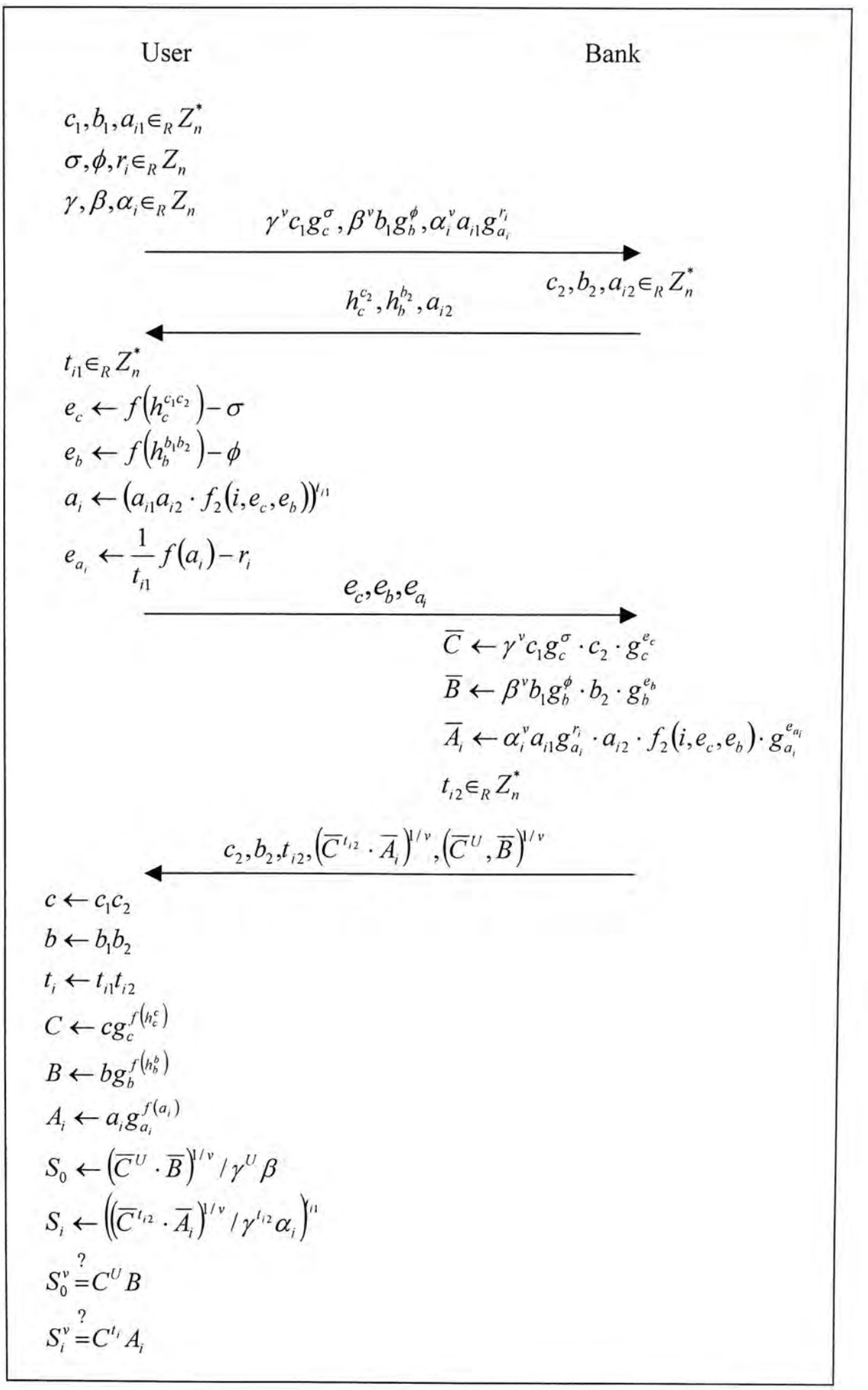

Figure 7.5: the withdrawal protocol of e-check system with unconditional privacy

Please note that in the figure occurrences of i are assumed to be over the range $1,\ldots,k$. You can find that if we only use one denomination term, it will become the original withdrawal protocol, i.e. k=1.

Step 1 The user generates random numbers $c_1, b_1, a_{11}, \ldots, a_{k1}$ (her contributions to the base numbers), $\sigma, \phi, r_1 \ldots, r_k, \gamma, \beta, \alpha_1, \ldots, \alpha_k$ (the blinding factors). And sends $\gamma^v c_1 g_c^{\sigma}, \beta^v b_1 g_b^{\phi}, \alpha_1^v a_{11} g_{a_1}^{r_1}, \ldots, \alpha_k^v a_{k1} g_{a_k}^{r_k}$ to the bank.

Step 2 The bank chooses her k+2 contributions to the base numbers $c_2, b_2, a_{12}, \ldots, a_{k2}$. And sends $h_c^{c_2}, h_b^{b_2}, a_{12}, \ldots, a_{k2}$ to the user. Sending $a_{12}, \ldots, a_{k2}$ directly allows user to raise one of the resulting signatures to a power she chooses.

Step 3 The user chooses k random numbers $t_{11}, \ldots, t_{k1}$ and computes

$$e_c \leftarrow f\left(h_c^{c_1 c_2}\right) - \sigma$$
$$e_b \leftarrow f\left(h_b^{b_1 b_2}\right) - \phi$$
$$a_1 \leftarrow \left(a_{11} a_{12} \cdot f_2(1, e_c, e_b)\right)^{t_{11}}$$
$$\vdots$$
$$a_k \leftarrow \left(a_{k1} a_{k2} \cdot f_2(k, e_c, e_b)\right)^{t_{k1}} \text{ where } f_2 \text{ is an one - way function}$$
$$e_{a_1} \leftarrow \frac{1}{t_{11}} f(a_1) - r_1$$
$$\vdots$$
$$e_{a_k} \leftarrow \frac{1}{t_{k1}} f(a_k) - r_k$$

The user sends $e_c, e_b, e_{a_1}, \ldots, e_{a_k}$ to the bank.

Step 4 The bank computes the blinded versions of $A_1, \ldots, A_k$, B and C.

$$\overline{C} \leftarrow \gamma^v c_1 g_c^{\sigma} \cdot c_2 \cdot g_c^{e_c}$$
$$\overline{B} \leftarrow \beta^v b_1 g_b^{\phi} \cdot b_2 \cdot g_b^{e_b}$$
$$\overline{A_1} \leftarrow \alpha_1^v a_{11} g_{a_1}^{r_1} \cdot a_{12} \cdot f_2(1, e_c, e_b) \cdot g_{a_1}^{e_{a_1}}$$
$$\vdots$$
$$\overline{A_k} \leftarrow \alpha_k^v a_{k1} g_{a_k}^{r_k} \cdot a_{k2} \cdot f_2(k, e_c, e_b) \cdot g_{a_k}^{e_{a_k}}$$

The following relations hold between the blinded numbers and their unblinded values:

$\overline{C} = \gamma^v C, \overline{B} = \beta^v B, \overline{A}_1 = \alpha_1^v A_1^{1/t_{11}}, \ldots, \overline{A}_k = \alpha_k^v A_k^{1/t_{k1}}$

The bank then chooses k random numbers $t_{12},\ldots,t_{k2}$ and sends $c_2, b_2, t_{12},\ldots,t_{k2}, \left(\overline{C}^{t_{12}} \cdot \overline{A}_1\right)^{1/v}, \ldots, \left(\overline{C}^{t_{k2}} \cdot \overline{A}_k\right)^{1/v}, \left(\overline{C}^U, \overline{B}\right)^{1/v}$ to the user.

Step 5 Using c_2 and b_2 the user can compute c and b as c_1c_2 and b_1b_2 respectively. She now constructs the number $C, B, A_1,\ldots,A_k$ as

$C \leftarrow c g_c^{f(h_c^c)}, B \leftarrow b g_b^{f(h_b^b)}, A_1 \leftarrow a_1 g_{a_1}^{f(a_1)}, \ldots, A_k \leftarrow a_k g_{a_k}^{f(a_k)}$

where $t_1 \leftarrow t_{11}t_{12},\ldots,t_k \leftarrow t_{k1}t_{k2}$.

And computes k+1 signatures

$S_0 \leftarrow \left(\overline{C}^U \cdot \overline{B}\right)^{1/v} / \gamma^U \beta$

$S_1 \leftarrow \left(\left(\overline{C}^{t_{12}} \cdot \overline{A}_1\right)^{1/v} / \gamma^{t_{12}} \alpha_1\right)^1$

$\vdots$

$S_k \leftarrow \left(\left(\overline{C}^{t_{k2}} \cdot \overline{A}_k\right)^{1/v} / \gamma^{t_{k2}} \alpha_k\right)^1$

Finally the user verifies the signatures she received by

$S_0^v \overset{?}{=} C^U B$

$S_1^v \overset{?}{=} C^{t_1} A_1$

$\vdots$

$S_k^v \overset{?}{=} C^{t_k} A_k$

The user ends up with $3k$+5 numbers γ, β, c, b, $a_1,\ldots,a_k$, $t_1,\ldots,t_k$ and $S_0,\ldots,S_k$ which are the 2 blinding factors, k+2 base numbers, k random parameters for the secret sharing line and k+1 signatures. The bank will list $\overline{C}, \overline{B}$ (where $\overline{C} = \gamma^v C, \overline{B} = \beta^v B$) and unique check ID U (we assume that U is concatenation of User's ID and check's ID) with user's account in case that user will refund the check later.

7.4.2 The Payment Protocol

Compare with withdrawal protocol, the payment protocol is much simpler (figure 7.6). Without loss of generality, we assume that the user wishes to spend an amount corresponding to $(g_{a_1},\ldots,g_{a_j})$, i.e. \$$2^j$-1 ($1 \le j \le k$), as example.

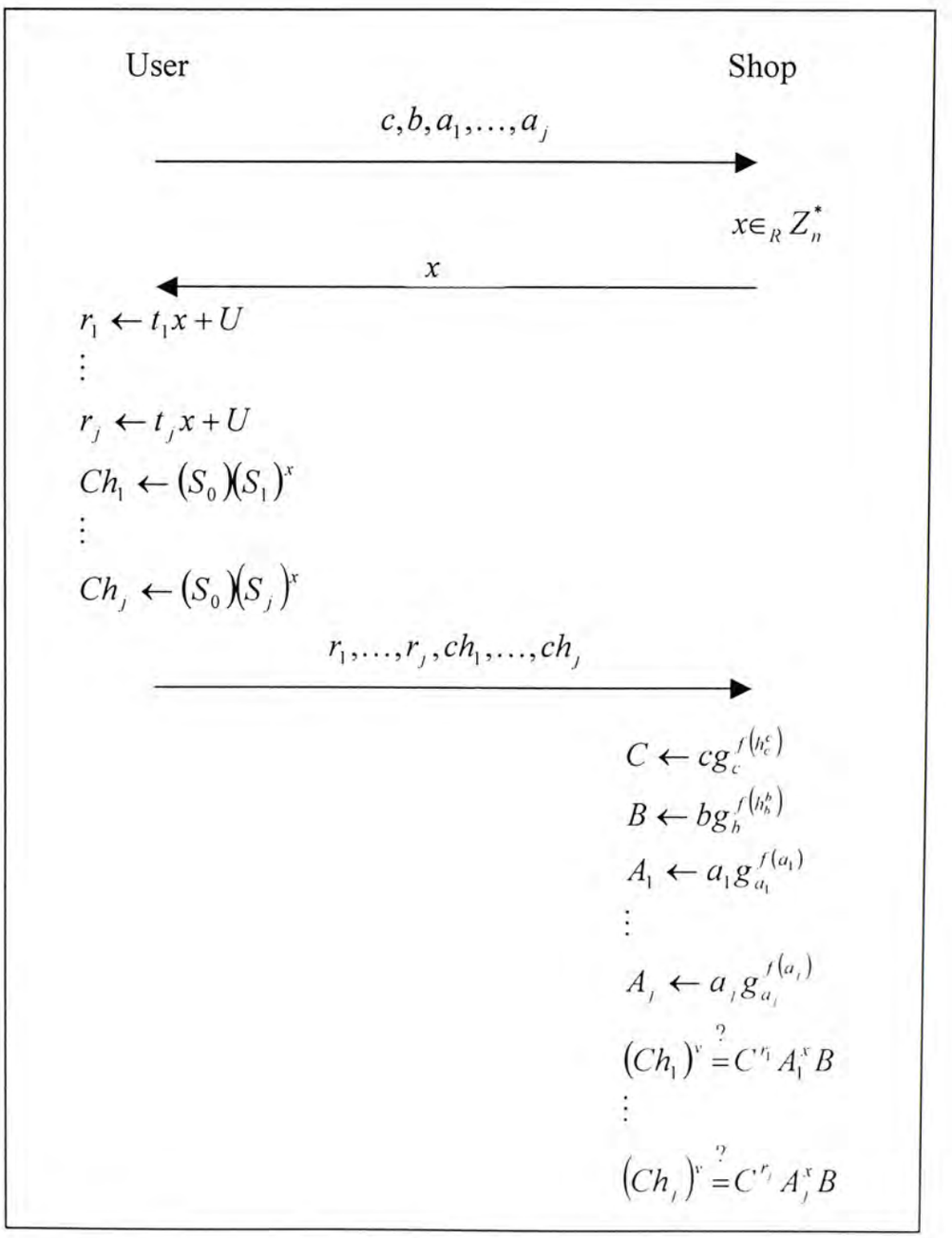

Figure 7.6: The payment protocol of e-check system with unconditional privacy

Step 1 The user starts by sending c, b, $a_1,\ldots, a_j$, to the shop.

Step 2 The shop replies a random challenge x.

Step 3 The user computes out $r_1,\ldots,r_j$ where $r_i=t_ix+U$ and sends them to the shop together with RSA signature that proves the response is correct, i.e. $Ch_i=(S_0)(S_i)^x$.

Step 4 The shop computes out j+2 base numbers

$$C \leftarrow cg_c^{f(h_c^c)}$$

$$B \leftarrow bg_b^{f(h_b^b)}$$

$$A_1 \leftarrow a_1 g_{a_1}^{f(a_1)}$$

$$\vdots$$

$$A_j \leftarrow a_j g_{a_j}^{f(a_j)}$$

and verifies j signatures by following methods

$$(Ch_1)^y \stackrel{?}{=} C^{r_1} A_1^x B$$

$$\vdots$$

$$(Ch_j)^y \stackrel{?}{=} C^{r_j} A_j^x B$$

The shop save signatures together with c, b, a_1, ..., a_j, x, r_1, ..., r_j for later deposit.

We still need to make a few addition to this protocol to make sure that the challenges are different at each payment time, the challenges can be generated by using an one-way function with unique inputs, for example $x=f$(shop-ID, time).

7.4.3 The Deposit Protocol

The deposit protocol is as simple as the deposit process of a coin.

Step 1 The shop sends c, b, $a_1,\ldots,a_j$, x, $r_1,\ldots,r_j$, $Ch_1,\ldots,Ch_j$ and the amount of check (i.e. $\$2^j$-1) to the bank.

Step 2 The bank verifies the signatures by following methods

$$C \leftarrow cg_c^{f(h_c^c)}$$
$$B \leftarrow bg_b^{f(h_b^b)}$$
$$A_1 \leftarrow a_1 g_{a_1}^{f(a_1)}$$
$$\vdots$$
$$A_j \leftarrow a_j g_{a_j}^{f(a_j)}$$
$$(Ch_1)^y \stackrel{?}{=} C^{r_1} A_1^x B$$
$$\vdots$$
$$(Ch_j)^y \stackrel{?}{=} C^{r_j} A_j^x B$$

If the check is valid, the bank increases shop's account by $\$2^j+1$.

Step 3 The bank then checks $a_1,\ldots,a_j$ in the checklist, to see if it has been double-spent before. The double-spender can be identified easily from different challenges and responses of double-spent checks.

7.4.4 The Refund Protocol

When the user wants a refund for the amount of the check which she did not spent, The user first informs the bank of his account number and the unique check ID

The bank verifies that U is listed with user's account and finds corresponding $\overline{C}$, $\overline{B}$.

Step 1 The user sends $a_{j+1},\ldots,a_k, t_{j+1},\ldots,t_k$ to the Bank.

Step 2 The bank searchs through his checklist to see if $a_{j+1},\ldots,a_k$ have been used before. If not, the bank random generates a number x, and sends it to the user.

The bank computes out $r_{j+1},\ldots,r_k$, where $r_i=t_i x+U$.

Step 3 The user does the same computation, find $r_{j+1},\ldots,r_k$ and then computes out $Ch'_{j+1},\ldots,Ch'_k$, where $Ch_i' \leftarrow (S_0)(S_i)^x \gamma^{r_i} \beta$.

Step 4 The bank verifies Ch'_i by $(Ch_i')^y \stackrel{?}{=} \overline{C}^{r_i} A_i^x \overline{B}$.

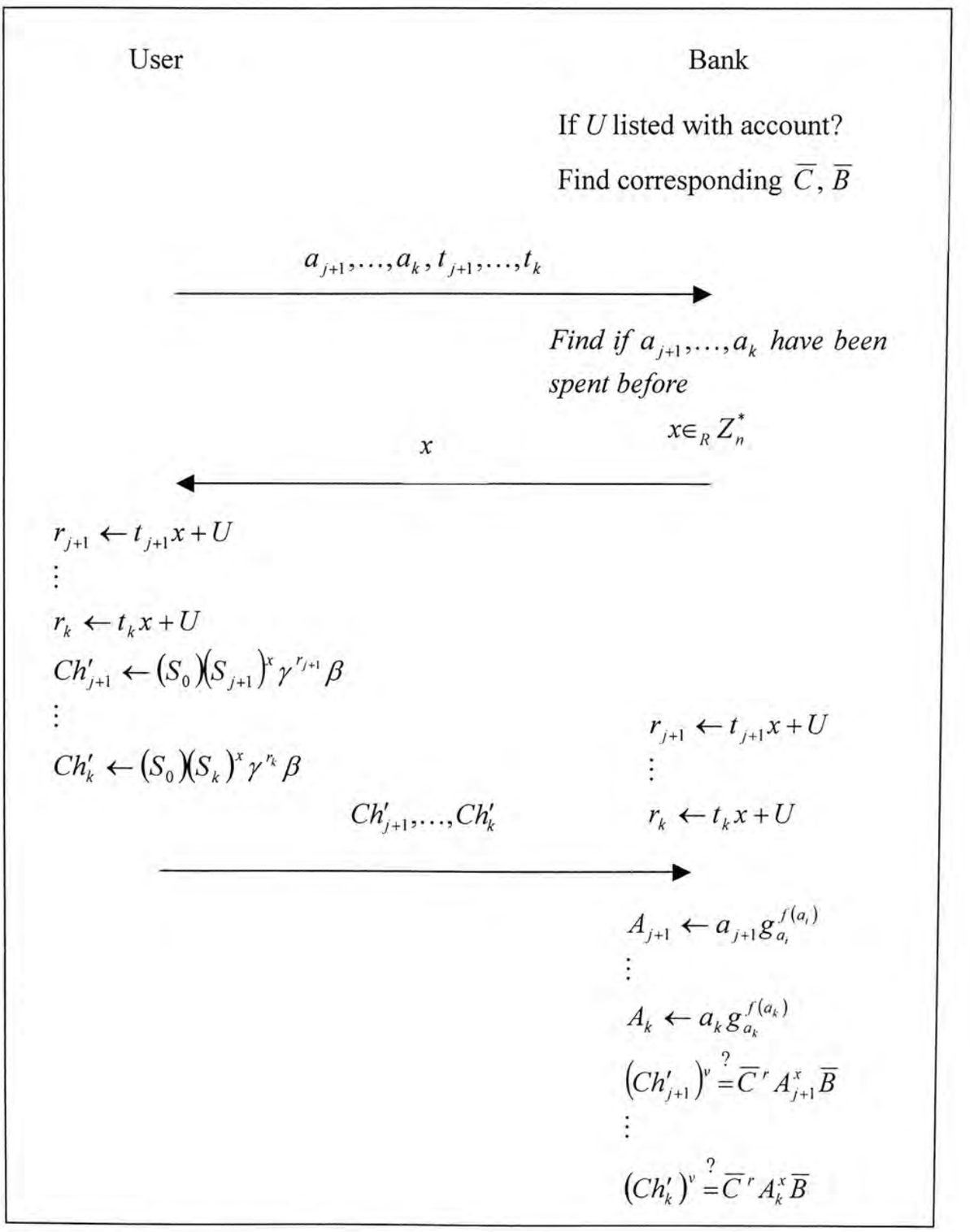

Figure 7.7: The Refund Protocol of e-check system with unconditional privacy

Step 5 The bank checks the checklist to see if $a_{j+1},\ldots,a_k$ have been used before. If not, the bank accepts the check and refunds the appropriate amount of money to user's account. And puts $a_{j+1},\ldots,a_k$ on refund list. She then erases U, $\overline{C}$, $\overline{B}$ from User's account.

7.4.5 Protocol Discussion

Privacy

In this protocol, we can found that the information we given in the payment doesn't link any information used in refund protocol. Bank cannot link the check that used in the payment and its refund together. It is easy to prove that the privacy of the user is guaranteed unconditionally.

Storage Space

For a maximum $\$2^k$-1 amount of check, the user must store 2 blind factor, k+2 base numbers, k secret values, 1 check ID and k+1 signatures. When k is large, compare to store $\$2^k$-1 amount of coin, it saves a lot of storage space. Moreover, we can smaller the size of check, if the bank signs different denomination term with different key. The user can multiply the terms together and separates them later by using the method described in section 5.2. But this method need more computational power to separate the signatures when the user want to pay the check to shop.

Computational Power

The computational requirements are much higher for an e-check system in the amount of payment that can be made by small number of coins. But the protocol is efficient when amounts that can only be paid with many coins are involved.

Extend to Transferability

We can include transferability (or denomination expansion) to e-check system by

using the method described in chapter 6. But it will increase systems complexity since user not only need to check each denomination terms but also need to verify all the transactions have made before during each payment.

7.5 Conclusion

In this chapter, we introduce two e-check systems. One is almost as efficient as a single coin system, but only support partial privacy. The bank only does 2 signatures during withdrawal, and refund protocol only need one communicaion.

The other e-check system guarantee both privacy and untraceability. The protocol needs more computational power than the previous one, but it is much more efficient compare with traditional e-check system which use multiple terms for random challenge and response.

Chapter 8
Conclusion

Paying for goods and services electronically becomes more and more common in our daily life. Since the late 1970s, a variety of schemes have been proposed to allow payment to be effected across a computer network

In this thesis, we have made a thorough study on electronic payment system by introducing the common models, showing their basic requirements, classifying schemes' efficiency, giving proposing history of some famous electronic payment schemes.

However most electronic payment schemes are very complex and slow to implement until the presentation of single-term payment system. In our thesis, we first introduced a single-term off-line coins system, which is the fastest to date among the proposed payment systems. And then, we present two new electronic payment systems: an off-line transferable e-coin system and an off-line e-check system. They are much more efficient and simpler than the old electronic payment schemes.

In off-line transferable e-coin system, user can spend the received coin in a later payment to a third person without contacting the bank or another central authority between two transactions. Before final user deposit the money, a list of user's information will be embedded in the coin. The bank can identify the double-spender and other user's privacy will be preserved.

In the e-check system, we constructed two protocols: one is very simple and efficient but only support partial user privacy; the other one support both privacy and untraceability but more complex to implement (it is still efficient compare with other

proposed e-check systems).

To conclude, electronic payment system will become more and more important because of information revolution. We believe that upon further modifications and development, electronic payment will gain a majority position in payment system of our coming life.

Reference

[Bra93] Stefan Brands. "An efficient off-line electronic cash system based on the representation problem". Technical Report CS-R9323, CWI (Centre for Mathematics and Computer Science), Amsterdam, 1993.

[Bra94] Stefan Brands. "Electronic cash systems based on the representation problem in groups of prime order". In proceeding of CRYPTO'93, 1994.

[CFN90] D. Chaum, A. Fiat, and M. Naor. "Untraceable electronic cash". In advances in Cryptology — Crypto '88 (Lecture Notes in Computer Science), pages 319-327. Springer-Verlag, 1990.

[CFT95] A. Chan, Y. Frankel, and Y. Tsiounis. "An efficient off-line electronic cash scheme as RSA". Research Report NU-CCS-96-03, northeastern University, Boston, Massachussets, 1995.

[CFT98] A.Chan, Y. Frankel, and Y. Tsiounis. "Off-line electronic cash as secure as RSA", 1996. Unpublished manuscript.

[CEvdG88] D.Chaum, J.H. Evertse, and J. van de Graff. "Demonstrating possession of a discrete logarithm without revealing it". In Advances in Cryptology – EUROCRYPT'87, pages 127-141, Berlin, 1988. Springer-Verlag.

[Cha83] D. Chaum. "Blind Signatures for Untraceable Payments". In Advances of Cryptology: Proceedings of CRYPTO '82, Plenum, NY, 1983, pages 199-203.

[Cha85] D. Chaum. "Security without Identification: Transaction Systems to make big Brother Obsolete". In Communications of the ACM, Vol.28,

No. 10, Oct. 1985, pages 1030-1044.

[Cha90] D. Chaum. "Online cash checks". In J.-J. Quisquater and J. Vandewalle, editors, Advances in Cryptology-EUROCRYPT '89, Lecture Notes in Computer Science, pages 288-293. Springer-Verlag, 1990.

[Cha92] D. Chaum. "Randomized blind signature". Personal communications, April 1992.

[ChaDB90] D. Chaum, B. den Boer. "Efficient Offline Electronic Checks". In Advances in Cryptology---Eurocrypt '89, 1990.

[ChaPe93] D. Chaum, T. Pedersen. "Transferred Cash Grows in Size". In Advances in Cryptology --- Eurocrypt '92, Proceedings, Lecture Notes in Computer Science, pages 390--407. Springer-Verlag, 1993.

[Cos90] M. Coster. "Some algorithms on addition chains and their complexity", TR CS-R9024, Centrum voor Wiskunde en Informatica, Amsterdam (June 1990)

[CraPe93] R. Cramer and T. Pedersen. "Improved privacy in wallets with observers". In advances in Cryptology: Eurocrypt '93, Proceedings (Lecture Notes in Computer Science 765), pages 329-343. Springer-Verlag, 1993.

[Dam90] I. B. Damgård. "Payment systems and credential mechanisms with provable security against abuse by individuals". In Advances in Cryptology, Proc. Of Crypto '88 (Lecture Notes in Computer Science), pages 328-335. Springer-Verlag, 1988. Berlin, 1990.

[Fer93] Niels Ferguson. "Single Term Off-Line Coins". In Advances in Cryptology-EUROCRYPT'93, Lecture Notes in Computer Science, Springer Verlag, 1993

[FY92] M. Franklin and M. Yung. "Towards provably secure efficient electronic cash". Technical Report: TR CUCS-018-92, April 1992.

[FY93] M. Franklin and M. Yung. "Secure and efficient off-line digital money". In Proceeding of the twentieth International Colloquium on Automata,

Languages and Programming (ICALP 1993), (Lecture Notes in Computer Science 700), pages 265-276. Springer-Verlag, 1993. Lund, Sweden, July 1993.

[MOV96] a. Menezes, P. Oorschot and S. Vanstone. Handbook of Applied Cryptography, CRC, 1996.

[Oli81] J. Olivos. "On vectorial addition chains", Journal of Algorithms 2 (June 1981), pages 13-21.

[OO90] T. Okamoto and K.Ohta. "Disposable zero-knowledge authentication and their applications to untraceable electronic cash". In Advances in Cryptology – proceedings of CRYPTO 89, pages 481 – 496, 1990.

[OO92] T.Okamoto, and K.Ohta "Universal Electronic Cash", In J. Feigenbaum, editor, Advances in Cryptology – proceedings of CRYPTO 91, Lecture Notes in Computer Science, pages 324 – 337. Spring-Verlag, 1992.

[PW92] B. Pfitzmann and M. Waidner. "How to break and repair a 'provably secure' untraceable payment system". In J. Feigenbaum, editor, Advances in Cryptology, Proc. Of Crypto '91 (Lecture Notes in Computer Science 576), pages 338-350. Springer-Verlag, 1992.

[Sch96] Bruce Schneier. "Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C". John Wiley & Sons, Inc., 1996.

[Sha79] Adi Shamir. "How to share a secret". In Communications of the ACM, 22(11):612-613, 1979.

[Sha83] Sdi Shamir, "On the Generation of Cryptographically Strong Pseudorandom Sequences", ACM Trans. On computer Systems, 1 (1983) pp38-44

[Sti95] Douglas R. Stinson. "Cryptography: theory and practize". CRC Press Inc., 1995.

[Tsi97] Y. Tsiaunis, "Efficient electronic cash, new Notions and techniques", Ph.t Thesis 1997, http://www.ccs.neu.edu/home/yiannis/pubs.html

[RSA78] Ronald Rivest, Adi Shamir, and Leonard Adieman. "A method for obtaining digital signatures and public-key cryptosystems". In Communications of the ACM, 21:120-126, February 1978.

[VA90] Hans van Antwerpen. "Off-line electronic cash". Master's thesis, Eindhoven University of Technology, department of Mathematics and Computer Science, 1990.

[Yao82] A. C. Yao. "Protocols for secure computations". In 23rd Annual Symp. on Foundations of Computer Science (FOCS), pages 160-164. IEEE Computer Society Press, 1982.