A MOBILE AGENT CLONE DETECTION SYSTEM USING GENERAL TRANSFERABLE E-CASH AND ITS SPECIFIC IMPLEMENTATION WITH FERGUSON'S E-COIN

BY

LAM Tak-Cheung

A THESIS

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF PHILOSOPHY

DIVISION OF INFORMATION ENGINEERING

THE CHINESE UNIVERSITY OF HONG KONG

MAY 2002

Acknowledgements

I would like to express my most heart-felt gratitude to my supervisor, Prof. Victor Keh-Wei Wei, who gave me the best instructions, whether in the knowledge area of my academic research or in writing and presentation skills, during my two years of study. It was his encouragement that made me hurdle every obstacle in my M. Phil career. I am every bit proud of being his student.

I must also thank every member of the Information Integrity Laboratory, for sharing in my countless moments of joy and distress. We lived the past two years in mutual support, together overcoming hardship, and I truly cherish this friendship. I especially thank Sandy Wong for patiently introducing me to e-cash in my early research, and also Chun Man and Ivan Ng for their inspiration in the mobile agent and hence my thesis idea now.

Jeff Cheng and Joseph Liu gave me precious advice in my late research. Simon Chan experienced the worst time of difficulty with me, and that is something I will never forget. Anthony Chan, Shum Kwan, Tao Yu and Bruce Tong always ate and hung out with me, and brightening chats with them have altogether made lab life worth living and much more colorful. I have not forgotten to thank the group of FYP students – they are intelligent students that brain storm very quickly and often led me to new discoveries just by discussing with them.

Last but not least, I have to thank my family and Cecy Kou. I could not have concentrated so well in my research had it not been their taking care of my living necessities.

Abstract of thesis entitled:

Mobile agents provide a new paradigm for the distributed computing environment. In this paradigm, an autonomous program, namely the *mobile agent*, migrates from host to host to achieve its designed goals. With its advantages in mobility and autonomy, it is suitable for the development of some applications such as e-commerce, the information retrieval systems, etc. However, security threats still cause the main deployment bottleneck for the mobile agent paradigm.

Although there are a number of protection schemes for mobile agents, not all of the security requirements have been reached. One of the security problems is the unauthorized cloning of mobile agents. Mobile agents can be copied with ease. Multiple instances of the same agent cause unintended extra transactions. On the other hand, the host can repudiate the transaction by falsely claiming the existence of clones.

In this thesis, we propose a system for the detection of the unauthorized cloning of mobile agents by malicious hosts. This system can detect clones after the occurrence of the unauthorized cloning and identify the culprit, which can then be handed in for penalty. The agent migration protocol is offline (non-centralized). The itinerary of an agent is privacy-protected. The main technique is to map the clone detection problem to the double-spending problem in the transferable e-cash, and then adopt and adapt the existing solutions from the latter field.

Our system is motivated by Chaum-Pedersen's general transferable e-cash model [ChaPe93]. By employing the general transferable e-cash model, we can apply a large class of the existing e-cash systems to our clone detection system. As an offline scheme, it lowers the communication overheads. However, it suffers from the delayed halting of malicious actions from agent clones.

Particularly, we present the implementation of the above general clone detection scheme with Wong's transferable extension [Won01] of Ferguson's single-term offline untraceable e-cash [Fer93]. In this specific implementation, we mimic the paper passport in real life, where itinerary records are chopped to the passport when a person moves from one country to another country. The records inside the agent's passport could be used for the investigation of any related cloning activities.

Submitted by LAM Tak-Cheung
for the degree of Master of Philosophy in Information Engineering
at The Chinese University of Hong Kong in May 2002

摘要

流動代理爲分佈式計算環境帶來了新的模範。在這模範裡，一個稱爲「流動代理」的自動化程式，由一個寄主遷移到另一個寄主，以達到她所設計的目標。她擁有流動性及自動化的好處，適合於發展電子商貿及資訊檢索等方面之應用。然而，安全性的威脅造成了流動代理模範發展的主要瓶頸。

雖然流動代理有一些保安的方案，但仍未能達至所有保安的要求。其中一個保安問題是未經授權的流動代理複製。流動代理是易於被複製的。來自同一代理本體的多個個體造成了有違意願的額外交易。另一方面，寄主卻可以假作聲稱複製代理的存在，作爲抵賴交易的藉口。

在此一論文中，我們建議了一個系統，用作偵測來自惡意寄主未經授權的流動代理複製。這個系統可以於事後偵測代理的複製，鑒別出犯罪者的身份，予以懲罰。代理遷移協定是離線的(非中央管理的)。代理路徑的私隱得以受到保護。當中的主要技術，是將流動代理的複製問題，投射到可轉移的電子貨幣的雙重消費問題上，並應用後者現有的解決方法。

我們的系統的動機，引發自 Chaum 及 Pedersen 的廣泛的、可轉移的電子貨幣系統。利用這個廣泛的、可轉移的電子貨幣模型，我們可以應用大量現有的電子貨幣系統於我們的複製偵測系統上。作爲一個離線系統，它減低了通訊的額外成本。然而，它延遲了制止複製代理的惡意行爲。

我們個別地利用了 Wong 所延伸至可轉移的 Ferguson 的一次性、離線、不可追蹤的電子貨幣，來介紹我們以上廣泛的代理複製偵測系統的實作應用。在這個個別的實作應用中，我們模仿了現實生活中的紙製護照，當一個人由一個國家去到另一個國家時，路徑的紀錄會被印於護照中。代理的護照中的紀錄可用作偵測有關的複製活動。

Contents

Chapter 1

Introduction

The meaning of the word *agent* is multi-fold.

In real life, we have different kinds of agents, such as the real estate agent, the traveling agent, the insurance agent, etc. Although these agents provide different natures of services, they are all authorized representatives working for its clients. The clients give a number of orders to the agents when they hire the agents. With certain authorization, the agents can make decision on behalf of their clients and do the rest of the task for them.

In some movies, the agent is modeled in another sense, which is full of mystery and intelligence. The FBI Agent - Agent Fox Mulder and Agent Dana Scully in the film *The X-files* and the Secret Agent 007 – *James Bond*, are typical examples.

In the digital world, *software agent* is an autonomous program, which simulates the human-like intelligence and behaviors of the agent in the physical world. It works independently on behalf of its owner and is distributed among heterogeneous host computers to achieve its designed goals.

Mobile agent is a special class of the software agent, which migrates from host to host, operating under its own control, with the resources maintained by the visited hosts, to attain its objectives. The movie *The Matrix* gave us a personification of the agent both in the physical world and the digital world. The agent in this movie is autonomous, mobile, reactive and able to learn. In this thesis, we concentrate on the security domain of the mobile software agent.

In this chapter, we give an overview for different aspects of the mobile agent. First, we present the history of development from the traditional computing models to the mobile agent paradigm. Then we show the potential benefits for different applications using this new computing paradigm. After that, security issues of mobile agent are briefly discussed. Lastly, we outline the organization of the thesis.

1.1 Evolution of the Mobile Agent Paradigm

Over the past few decades, the computing technology has been undergoing a drastic growth.

Before the Internet era, computations solely rely on bulky standalone devices. These devices are sophisticated enough to process all the computations as required at that time. Users can then retrieve the processed data through dumb terminals.

Advanced by the fast deployment of the computer network, computations are no more restricted to several dedicated devices. The centralized approach is abandoned. By then, a number of important applications and technologies have developed under the distributed computing model. In chronological order, they are the message passing systems, remote procedure call, distributed object systems and mobile agent systems, where the last one is built on top of the previous one.

The basic communication between two parties over the network involves the exchange of simple passive messages. This forms the core of the message passing systems, such as the Web, the email and the FTP. The device which provides the services is called the server, while the one which obtains the services is called the client. The simple message passing system is fundamental. However, it limits the clients from expanding the capabilities of the servers. To enhance the flexibility, the remote procedure call is applied.

In the remote procedure call (RPC), the server exposes bundles of software modules to clients. The program in the client performs function calls from the shared modules at the server. The function call at the server is transparent as it is a local function call at the client side.

In distributed object systems, instead of making function calls to the server, the client invokes the pre-defined objects residing on the server. The client's program can access the object's properties and methods through the object interface. The examples of distributed object systems include CORBA and DCOM.

In RPC and distributed object systems, the server can expand its services by adding new functions or objects. However, there is still no way for the client to customize the given functions and objects. Therefore, the concept of mobile agent is introduced for the customization of software modules.

With mobile agent, we no longer request the functions or objects from the remote site. Instead, the client sends directly a self-contained entity (an autonomous program) to the server for computations and resource allocation. The machine or device that provides the working environment for the mobile agent is called the *host*. The client machine which creates the mobile agent is called the *owner host*. The owner host could customize its mobile agent for specific designed goals.

The typical life cycle of a mobile agent can be described as below:

First, the mobile agent is created and resides in the owner host. It is then dispatched to another host for execution. During the execution, it collects certain information from the accommodating host, and generates runtime states. It suspends its execution at the hosting machine and transfers itself to the next host. It then resumes its execution according to the runtime states on the new host. The execution-migration process continues until the mobile agent returns to the owner host, with its task completed.

Although a concise definition has not yet been established for the mobile agent, it is generally agreed that a mobile agent should be at least autonomous, adaptively learning, and mobile [Sun98]. To be autonomous, the mobile agent should be able to work independently on behalf of its owner under its own control. By adapting and learning, the mobile agent's action is directed by its goals, the past experience and the present working environment. Being mobile, the mobile agent moves around distributed hosts for executions.

More properties are described in [FG96] to differentiate mobile agents from other traditional software pieces, such as persistency, goal-oriented capability, communicative and collaborative power, etc.

The mobile agent paradigm spots a new area for researchers to explore. A number of applications, such as the electronic commerce, the information retrieval systems and the workflow systems are benefits from this evolution. But at the same time, new problems are induced. Security problem is one of the most important issues in studying mobile agents, which is also the main focus of our paper.

1.2 Beneficial Aspects of Mobile Agents

The concept of mobile agents enhances a number of benefits for the distributed computing environment on top of the traditional client-server model. They have been discussed extensively in literatures [FPV98, HCK95, LO99]. The major advantages lie in autonomy, client customization, reduction of network traffic and latency, and asynchronous computation.

Autonomy: The added intelligence enables the mobile agent to work independently outside its owner host. Instead of asking the owner host for each decision making, the mobile agent decides for itself the reaction and the next destination based on the present situation and the past experience. This feature reduces the need for intervention from human users.

Client Customization: In the traditional RPC and distributed object systems, to extend the features or services, we install new functions or objects into the server. There is no room for the

client to customize and specify the operations of software pieces based on its own requirements. On the other hand, mobile agent is virtually a program, carrying code and states, installed and run in the remote host. Functionality is no longer confined to the pre-defined functions or objects offered by the server. Clients are free to implement their specific mobile agents to satisfy their own needs and interests.

Reduction of Network Traffic and Latency: Some applications in the traditional computing model require a large number of interactions between the client and the server. Resources are exchanged through the network for completion of computations. It is especially undesirable for a low bandwidth, high latency and high cost network. With mobile agents, we send the code directly to remote hosts. Computations are done locally and in real-time within the remote host. The mobile agent only returns necessary results to the owner host. The overall network traffic could be reduced if the size of code and data carried by the mobile agent is smaller than the resources required for computations [SS97].

Asynchronous Computation: Continued communications between the client and the server are not required. The workload for computations is shifted from the local host to the remote host, where the saved computation power could be re-allocated to other tasks. The asynchronous connection requires the host to get online only when the mobile agent is dispatching or arriving.

In spite of the above benefits, the use of mobile agents is still not as popular as the traditional distributed computing models. The main concern is on the security vulnerabilities generated by this new paradigm, which is an obstacle for the mobile agent development.

1.3 Security Threats of Mobile Agents

Every coin has two faces. While we appreciate the beautiful properties of the mobile agent paradigm, we suffer from the new flaws that did not exist in the old models.

Security threats are the fundamental deployment bottleneck for mobile agent. Attacks and defenses to the mobile agent system are widely studied in literatures [FGS96, Che98, JK99, NC99, GM00, Jan99]. The threats mainly fall into two categories: (1) Agent-to-Host attack, and (2) Host-to-Agent attack.

In the first class of attack, the hostile agent imposes harmful actions on the honest host. Like the traditional malicious program, such as Trojan horses, viruses and worms, the hostile agent may invade the honest host by masquerading, denial of service, or unauthorized access [JK99]. Since the protection mechanisms for traditional malicious programs have been identified in early times before the mobile agent paradigm, similar techniques such as

authentication, access monitoring, audit logging, etc, can be used to secure the executing host. More about the countermeasures to the agent-to-host attack are discussed in [NL98, BGS98, Fon99].

In the second class of attack, the mobile agent is abused by malicious hosts. Incursion methods include eavesdropping of the agent's communicating conversation, alteration of agent's code and data, unauthorized access of agent's internal secret, or simply refusing of agent's execution. To reach a satisfactory protection scheme, several security requirements are listed [JK99]. They are integrity, anonymity, confidentiality, accountability and availability.

Integrity: The mobile agent should be protected from any unauthorized modification to the code, states, or any information carried by the agent. The agent itself cannot prevent a malicious host from the above tampering, but we can take measures to detect it and trace it back to the offending host.

Anonymity: In the mobile agent system, not only is the identity of the agent owner hidden, but also the hosts it has visited along the itinerary. We have to balance between the agent's need for itinerary privacy and the host's need to account for any agent's illegal actions.

Confidentiality: Any unauthorized access to the private data of the mobile agent is prohibited. Although the host can execute the agent, it is not able to interpret the decision logic and the execution flow behind the code.

Accountability: For any malicious actions done on the mobile agent, we can record the evidence and give the proof for non-repudiation purpose. The record serves as the after-the-fact logging to trace and identify any malicious parties.

Availability: The protection mechanism should ensure that the host's resources, such as network connectivity, file access, and code library are allocated to the mobile agent fairly and faithfully. Otherwise, the mobile agent fails to achieve its designed objectives due to inadequate and inappropriate resource allocation.

Since the resources and the computation power are assigned by the host, the host takes overall control of the mobile agent. This unbalanced relationship makes the mobile agent more vulnerable to the malicious host's attack. More research works are carried out for the sake of such host-to-agent attack [KAG98, BMW98, Hoh98, ST98, NC99]. Partial solutions, including network management and cryptographic techniques, are applied in dealing with these security threats. However, mobile agent security is not in its perfection. Open issues such as interpreter tampering, incomplete execution, agent kidnapping, agent cloning still remain unsolved [FGS96], or at least difficult to solve.

In particular, we explore one of the security areas in this thesis, where a countermeasure for mobile agent cloning is discussed. Cryptographic techniques are used to account for any unauthorized cloning of mobile agent from malicious hosts. Offending hosts are identified and

penalized.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, fundamental cryptographic theories are reviewed.

In Chapter 3, we discuss the security problems of unauthorized cloning of mobile agents. The countermeasure from Baek [Bae98] is also introduced in this chapter. Chapter 4 gives the details of several e-cash systems, which are essential for constructions of our mobile agent clone detection system in subsequent chapters.

Our mobile agent detection system using Chaum-Pedersen's general transferable e-coin [ChaPe93] is presented in Chapter 5. Specific implementation with transferable version of Ferguson's e-coin [Fer93, Won01] is described in Chapter 6.

Lastly, we draw conclusions in Chapter 7.

Chapter 2

Background of Cryptographic Theories

2.1 Introduction

Cryptographic systems have been employed for centuries by military and diplomatic parties for keeping secrets and for authentication. The intelligence in information exchange has sometimes decided the results of battles or even a war. With the booming of the computer and Internet, cryptography is not only a weapon between countries in the physical world, but it also plays important roles in the digital world. Over the last decade, cryptology has advanced from art to science and it is incurred drastic evolution with the fast-growing global communications and commercial applications. One of the applications nowadays is the mobile agent system, where dozens of cryptographic techniques and methodologies are hired to achieve certain security requirements.

In this chapter, we briefly introduce some basic but essential cryptographic techniques necessary to the understanding of this thesis. Readers who are already familiar with this material may skip ahead to chapter 3 and refer back as required. Details of mathematics can be referred to textbooks [Sim92, CLS94, Sti95, Sch96, MOV97, KR98]

2.2 Encryption and Decryption

Encryption is a process to transform a readable message (called *plaintext*) into another format such that the original information could not be retrieved by an adversary. An encryption key is

used for this transformation. The encrypted message is called *ciphertext*.

Decryption is a reverse process of encryption. It transforms a ciphertext to its original plaintext with a decryption key.

The mathematical expressions of encryption and decryption processes are described below:

$$E_e(P) = C$$

$$D_d(C) = P$$

where E = the encryption function

D = the decryption function

P = the plaintext

C = the ciphertext

e = the encryption key

d = the decryption key

2.3 Six Cryptographic Primitives

At the basic level of cryptography, there are six cryptographic primitives. They are symmetric encryption, asymmetric encryption, digital signature, message, digital certification, and zero-knowledge proof. Many cryptographic applications are derived from different combinations of these six primitives.

2.3.1 Symmetric Encryption

In symmetric encryption schemes, in most of the cases, the encryption key is the same as the decryption key, and that is why it is so called *symmetric*. The communication parties should first make an agreement on the encryption/decryption key secretly before they can communicate securely. Otherwise, if some other else knows this key, then she can encrypt and decrypt the message. Therefore, the symmetric encryption is also called the secret key encryption and we call the encryption/decryption key a secret key.

The mathematical expressions of encryption and decryption processes are described below:

$$E_s(P) = C$$

$$D_s(C) = P$$

where E = the encryption function

D = the decryption function

P = the plaintext

C = the ciphertext

s = secret key/encryption key/decryption key

Examples of symmetric encryptions include the Data Encryption Standard (DES) [NBS77, NIST88], the Triple DES [ANSI85] and the International Data Encryption Algorithm (IDEA) [Lai92].

2.3.2 Asymmetric Encryption

In asymmetric encryption schemes, the encryption key is the different from the decryption key, and that is why it is so called *asymmetric*. Furthermore, the decryption key cannot be derived from the encryption key in a reasonable amount of time. The receiver publishes the encryption key to a public directory so that the potential senders can use this key to encrypt the message. The receiver then uses the privately stored decryption key to decrypt the message. The encryption key is called the public key and the decryption key is called the private key. Therefore, the asymmetric encryption is also called the public key encryption.

The mathematical expressions of encryption and decryption processes are described below:

$$E_e(P) = C$$

$$D_d(C) = P$$

where E = the encryption function

D = the decryption function

P = the plaintext

C = the ciphertext

e = public key/encryption key

d = private key/decryption key

where $e \neq d$

The concept of asymmetric ciphers was jointly invented by Diffie and Hellman in 1976 [DH76]. Examples of asymmetric encryptions include RSA [RSA78] and Elliptic Curve Cryptosystem [Mil85, Kob87]. However, Schneier [Sch98] indicated that the concept of asymmetric ciphers was not an invention of Diffie and Hellman. Declassified UK secret documents showed that 3 UK intelligent officers (mathematicians by training) invented both the concept of asymmetric ciphers and the RSA implementation, two years earlier than the Americans.

2.3.3 Digital Signature

Handwritten signatures have always been used as a proof of authorship of, or at least agreement with, the contents of a document. Digital signature is designed for a similar purpose. However, instead of signing to a paper document, it is signed to an electronic document, with the following properties: [Sch96]

1. *The signature is authentic. The signature convinces the document's recipient that the signer deliberately signed the document.*
2. *The signature is unforgeable. The signature is proof that the signer, and no one else, deliberately signed the document.*
3. *The signature is not reusable. The signature is part of the document; an unscrupulous person cannot transfer the signature to a different document.*
4. *The signed document is unalterable. After the document is signed, it cannot be altered.*
5. *The signature cannot be repudiated. The signature and the document are physical things. The signer cannot later claim that he or she did not sign it.*

In public key cryptosystems, Alice could sign her document with her private key and send the signed document to Bob. Bob can then verify this signature with Alice's public key.

2.3.4 Message Digest

Message Digest is also known as the *fingerprint* produced by one-way hash functions. In the physical world, a man can easily give his own fingerprint. From the fingerprint itself, we cannot derive how this man looks like. However, since it is infeasible to find two different men who give the same fingerprint, we say the same fingerprint is originated from the same man with a reasonable assurance of accuracy. A one-way hash function serves similarly in the digital world.

A hash function is a function f which has, as a minimum, the following properties: [MOV97]

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.*

A one-way hash function (OWHF) is a hash function f with following additional properties:

1. Given output y, it is computationally infeasible to find an input x such that $y = f(x)$
2. Given input x, it is computationally infeasible to find another input $x' \neq x$, such that they produce the same output $f(x) = f(x')$.

A collision resistant hash function (CRHF) is a hash function f with following additional properties:

1. Given input x, it is computationally infeasible to find another input $x' \neq x$, such that they produce the same output $f(x) = f(x')$.
2. It is computationally infeasible to find any two distinct inputs x and x', which produce the same output $f(x) = f(x')$

It is generally assumed that the algorithmic specification of a hash function is publicly known. It could be used in producing digital signatures. Two famous hash functions are Secure Hash Algorithm-1 (SHA-1) and MD-5. The former produces a 160-bit hash-value while the latter produces 128-bit.

2.3.5 Digital Certificate

A digital certificate is similar to the ID card in the physical world, which is used for the authentication of one's identity. The certificate contains the personal information of its owner, the owner's public key, the information of Certificate Authority (CA) and CA's signatures on the above information. The Certificate Authority is a trusted third party, such as the Verisign, the Microsoft, and the post office in Hong Kong.

If Alice wants to communicate with Bob securely, she uses Bob's public key to encrypt the message she sent. To prevent Bob from being masqueraded by Carol, Alice requires Bob to show her his digital certificate instead of his public key. Then Alice uses CA's public key to verify the certificate's information. This ensures that the public key is from Bob instead of Carol. She then uses the public key shown on Bob's certificate to encrypt the message and sends it to Bob. After all, Bob uses his private key to decrypt the message received. Even if Bob is masqueraded by Carol, with a copy of Bob's digital certificate, Carol cannot decrypt the message from Alice because Carol does not have Bob's private key.

2.3.6 Zero-Knowledge Proof

In the physical world, a proof of knowledge requires disclosure of the knowledge itself in most cases. There is no secret about the knowledge once we are required to give a proof on it. In cryptography, we always have secrets, such as the private key, the user identity, etc. To show the knowledge on these secrets without revealing them, we use the technique of zero-knowledge proof. Zero-knowledge proof is also known as *challenge and response*, in which one side

randomly initiates a question while the other side offers the correct answer.

Following is one example of zero-knowledge protocol given by Chaum, Evertse, and Graff [CEvdG88].

1. The values A, B and p are publicly known while x is Alice's secret. Alice wants to show Bob that she know x without revealing it, where $A^x = B \pmod{p}$.
2. Bob sends a random challenge b to Alice.
3. Alice sends back a response $s = r + bx \pmod{p-1}$ and $h = A^r$, where r is a random number with $r < p$.
4. Bob verifies the response by finding that whether $A^s \pmod{p}$ equals to $hB^b \pmod{p}$. If yes, Bob trusts Alice of knowing x, and vice versa.

In the whole protocol, Bob does not know x, given A, B, p. He even cannot derive x from the received parameters s and h. However, he can judge whether Alice has the knowledge of x.

2.4 RSA Public Key Cryptosystem

The RSA Public Key Cryptosystem [RSA78] is a widely used algorithm in the modern cryptography. It is first introduced in 1978 by Rivest, Shamir and Adlemant to worth the name RSA. RSA is based on modular arithmetic and its security is based on the difficulty in factoring a very large number, composite by two large primes.

The mechanism of RSA public-key encryption is described as follow:

Key pair preparation:

1. Choose two large prime p and q of roughly the same size.
2. Compute $n = pq$
3. Pick a large prime e such that $\gcd(e, (p-1)(q-1)) = 1$
4. Find an integer d by Euclidean Algorithm such that

$$de = 1 \pmod{(p-1)(q-1)}$$

or equivalently

$$d^{-1} = e \pmod{(p-1)(q-1)}$$

Now we prepared the public key n and e, and the private key d.

Encryption: To encrypt plaintext M to cipher C with e, we computes

$$C = M^e \pmod{n}$$

Decryption: To decrypt cipher C to plaintext M with d, we computes

$$M = C^d \pmod{n}$$

Signing Document: To give signature σ on document M with d, we computes

$$\sigma = M^d \pmod{n}$$

Verifying signature: To verify the signature σ with e, we computes

$$M = \sigma^e \pmod{n}$$

The above equations hold because

$$\begin{aligned} & M^{de} \pmod{n} \\ = \; & M^{1+K(p-1)(q-1)} \pmod{n} \\ = \; & M\,(1)^{K(q-1)} \pmod{p} \quad = M\,(1)^{K(p-1)} \pmod{q} \quad \text{(By Fermat Little Law)} \\ = \; & M \pmod{pq} \qquad\qquad = M \pmod{n} \qquad\qquad \text{(By Chinese Reminder Theorem)} \end{aligned}$$

Therefore, the original message M is recovered.

2.5 Blind Signature

Blind signature is a class of digital signatures where the signer has no knowledge on the signed content. It was first proposed by Chaum [Cha82] in 1982, with the first RSA-based blind signature protocol in [Cha85]. The blind signature is a very important technique for the implementation of untraceable e-cash.

The concept of a blind signature can be illustrated by an example of a paper document. The paper analog of a blind signature can be implemented by wax sealed envelopes, containing a carbon paper covering the document for signing. Signing outside the sealed envelope leaves a carbon copy of the signature on the document, without knowing its content.

Take Chaum's implementation as an example. Suppose Alice requests Bob for a signature on the message M without disclosing it to Bob. Using RSA algorithm, Bob has a public key e, a private key d, and a public modulus n.

Step 1: Blinding - Alice randomly chooses a value k called *blinding factor* which satisfy $1 < k < n$ and $\gcd(n, k) = 1$. Alice computes the blinded message $M^* = k^e M \pmod{n}$ and sends M^* to Bob.

Step 2: Signing – Bob computes the blind signature $\sigma^* = (M^*)^d \pmod{n}$ and sends σ^* to Alice.

Step 3: Unblinding – Alice computes the unblinded signature of M by $\sigma = \sigma^*/k$ and the result should be $\sigma = M^d \pmod{n}$, that is, Bob's signature of M.

Since k is random, Bob cannot determine x from σ^*, while a valid signature σ from Bob is given to Alice.

2.6 Secret Sharing

The idea of secret sharing is to divide a secret into pieces, which shared among users such that the pooled secret shares of specific subsets of users allow reconstruction of the original secret.

One simple version of this scheme, called (*t, n*)-threshold scheme, is to divide a secret into *n* pieces, such that any *t* of them can be used to reconstruct the secret. The definition of this scheme is quoted as follow:

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 in which knowing only t-1 or fewer shares provides no advantage (no information about S whatsoever, in the information-theoretic sense) to an opponent over knowing no pieces.*

For details of other algorithms in secret sharing, please refer to [MOV96, Sha79, Bla79]

2.7 Conclusion Remarks

Cryptography is important in the issue of information security. We briefly introduced the six cryptographic primitives: symmetric encryption, asymmetric encryption, digital signature, message digest, digital certificate and zero-knowledge proof. Some fundamental cryptographic systems, including the RSA public key cryptosystem, the blinding signature scheme and the secret sharing scheme are also presented. They are essential for secure applications such as electronic payment systems and mobile agent systems. In later parts of this thesis, we base on the above cryptographic tools to build a new mobile agent clone detection system.

Chapter 3

Background of Mobile Agent Clones

3.1 Introduction

Mobile agents offer a new paradigm to the distributed computing environment. It is an autonomous program moving around the network to achieve its objectives on behalf of its owner. Once the mobile agent resides outside its owner host, the owner host no longer has full control on the agent. Therefore, mobile agents are vulnerable to the attack of malicious hosts. Various kinds of attacks and their defenses to mobile agent are studied [KAG98, BMW98, Hoh98, ST98, NC99]. One simple but powerful attack is merely by raw copy and replay of the mobile agent. This kind of attack is called *cloning*.

In this chapter, we first classify the types of agent cloning that we focus on in this thesis. Then we will demonstrate the problems of unauthorized agent cloning with a *buyer agent* example. Baek's agent clone detection system [Bae98] will be presented, which tries to solve these problems by monitoring the life cycle of each agent and granting permission for each agent's action by a Trusted Third Party (*TTP*).

3.2 Types of Agent Clones

There are two classes of agent cloning. They are *authorized agent cloning* and *unauthorized agent cloning*.

Some mobile agent systems, such as Aglet system [KLO97], support the agent clone generation facility. Some literatures even use this clone property for load balancing in distributed systems [She99, SSCJ98, Fan01]. Nevertheless, the clone in these systems has a different identifier from its original agent, although it has the same code and states as the original one at a given time. This class of cloning, with different identifiers, is regarded as an authorized cloning. The authorized clones could be distinguished from their identifiers.

Baek [Bae98] redefines a clone as the copied agent, which has the same code, states and also the identifier as the original one. Based on his definition, it is easy to make a clone without any knowledge on mobile agent systems, such as Aglet and Mole [KLO97, SBH96], causing security threats. This class of cloning with the same identifier is regarded as an unauthorized cloning. We cannot distinguish between two clones from each other.

In this paper, we focus on the unauthorized cloning of mobile agents.

3.3 Mobile Agent Cloning Problems

A number of literatures use a buyer agent example to elaborate the problems of unauthorized mobile agent cloning [JK99][Bae98]. The problems include unexpected multiple transactions, repudiation to agent's transactions and black-box testing by parallel executions. Here we use a similar example for the same purpose:

"A buyer agent is designed and created by its original host to travel from host to host to buy (say a cake) at the lowest price. However, it does not buy any cake with the price higher than $100. Only the buyer agent knows this price ceiling ($100), but not the shopper hosts. When the buyer agent visits a shopper host, the shopper host offers the price. The buyer agent compares the prices offered by different shopper hosts, and places order to the shopper host with the lowest price offered. Finally, it returns to the original host to report the order it placed."

Following are some possible security problems of cloning of the buyer agent:

Unexpected Multiple Transactions. If a shopper host clones the buyer agent, then both of the clones independently place their buying orders, all under the name of the original host. The original host will be charged with multiple orders and has to pay more than its originally intended amount.

Repudiation to Agent's Transactions. If the owner host wants to refuse the buying order, it can have an excuse by falsely claiming that the transaction is done by a malicious clone instead of its buyer agent.

Black-box Testing by Parallel Executions. The shopper hosts may try to extract the price ceiling information by interpreting the buyer agent. [Hoh98], et al, try to prevent such kinds of

interpretation by modeling an agent as a *black-box*. The attacker is able to execute the agent, but it is not able to determine the inner semantic mechanisms of the agent. However, the attacker can still test the protected information by parallel executions of agent clones and examine the outputs. The shopper host can initially set a higher price offer, and then decrease the price to each clone until the buyer agent is willing to place the buying order.

The security problems caused by unauthorized mobile agent clones are not limited to the above scenarios. As each host is independent and distributed, we cannot control each host and prevent a host from performing the agent cloning. Moreover, since unauthorized clones are exact duplicates, we cannot differentiate between an original agent and its copy. Therefore, instead of preventing it from happening and distinguishing between the original and the copy, the target is shifted to detecting the existence of clones and identifying the host that made the clones.

3.4 Baek's Detection Scheme for Mobile Agent Clones

Baek [Bae98], et al, proposed a mobile agent clone detection scheme. In his scheme, any unauthorized clones of agents are detected online, i.e. under supervisions of a Trusted Third Party (*TTP*) for each action of the agent. All actions of the agent are stopped immediately once clones are detected. The host that clones the agent is identified by the TTP. The system is modeled by a special graph called *Coloured Petri Nets (CP-Nets)* [Jen92a, Jen92b]. A correctness proof is also given based on this CP-Nets Model.

3.4.1 The Main Idea

In Baek's scheme, a mobile agent is required to request a *Token* from the TTP, whenever it is created, executed, moved or destroyed. The Token contains the information of the agent identifier, the host identifiers (source and destination), the Token type, the signature from the TTP, etc. The agent is permitted to perform the above four actions (creation, execution, movement, destruction) only when the corresponding type of Token is granted by the TTP.

Baek assumes that the expected life cycle of an agent (without cloning) should start by a creation, follow by multiple executions and movements, and finally end up by a destruction, i.e. (create, execute, move, execute, move, ... , move, execute, destroy). If the sequence of Token type requested by an agent does not match with the expected life cycle, then there is a clone. The TTP identifies the cloning host from the Token request history and stops the protocol by not issuing the requested Token to the agent.

The above idea is modeled by two sub-nets of the CP-Nets. The first sub-net models the Token request sequence (the real life cycle of the agent) and the second sub-net models the clone detection mechanism (the expected life cycle of the agent). The first sub-net issues a Token request to the second sub-net, while the second sub-net verifies and grants the Token to the first sub-net. The correctness proof of this model is also given in CP-Nets semantics.

3.4.2 Shortcomings of Baek's Scheme

Baek's scheme is an online scheme, in which every action of an agent is monitored by the TTP. An online scheme has an advantage that it can stop the protocol immediately once clones are detected to avoid further malicious actions by the clones. However, there are tradeoffs in employing an online detection scheme, as described below.

Communication Overhead. For every action (creation, execution, movement and destruction) of an agent, a Token request is sent to the TTP and a Token is granted to the host from the TTP, which results in a considerable communication cost.

Itinerary Privacy. TTP knows the sequence of actions done by an agent. It also knows where the agent has ever visited. Although the TTP is trusted, it is still undesirable for revealing such sensitive information, especially for the applications in the e-commerce.

Autonomy. One of the designed goals of mobile agent is *autonomy*, i.e. the agent has full control in decision making on its own actions without asking its owner host or other outside parties. The permissions from TTP limit the autonomy of agents to a certain degree.

Another issue about the storage of Tokens is addressed by Baek. If an agent clone is sent from the same source host to the same destination host, then a cloned *Movement Token* could be used for migration of the agent, without violating the clone detection mechanism. When the receiving host executes the agent clone, the clone detection mechanism will identify the receiving host as the cloning host due to multiple executions, instead of the sending host. Baek states that it is the responsibility of the receiving host to check that the receiving Movement Token is unique against the previous received Movement Tokens, i.e. each host has to store all Movement Tokens received for checking, which consumes more storage space as the number of visiting agents increases.

Similar problem exists in a cloned *Execution Token*. A host can perform black-box testing by parallel executions of agent clones with cloned Execution Token. The host sends away the agent clone with desirable tested output only. From the point of view of the TTP, there is only one Execution Token request from the host, and hence this cloning host is not identified.

3.5 Conclusion Remarks

We introduced an attack to mobile agents called cloning. We can attack it with ease by simple copying of the agent, which causes multiple instances to co-exist in the system. Since the unauthorized agent clones are exactly identical, including the identifier, we cannot distinguish between the copy and the original. A malicious host can make use of these clones to perform illegal activities, while a dishonest owner can repudiate any transaction by falsely claiming that a clone had made it.

Baek proposed a scheme to deal with the problems of agent cloning. Instead of preventing it from happening, his scheme detects the clones and identifies the culprit. It is an online scheme that requires centralized supervisions.

As we will see later, we propose another mobile agent clone detection system in this thesis. Our scheme is based on the cryptographic techniques of *general transferable e-cash.* The scheme is offline, with itinerary privacy.

Chapter 4

Background of E-cash

4.1 Introduction

E-cash is the electronic counterpart of the physical cash as a medium of payment in electronic transactions. In a *transferable e-cash*, the payment recipient can immediately use the received e-cash to pay another entity. E-cash is an active area of research in cryptography [CFN88, vA90, OO91, Fer93, Bra93, Won01].

E-cash is ultimately an electronic file, in special form, which is easy to make an exact copy. A user can have multiple uses of these copies and we call this abuse of e-cash *double spending*. One of the security requirements of e-cash is to combat against double spending. In many schemes, double spending cannot be prevented from happening. Rather, the e-cash system examines the circulations in the system and checks for duplicates. When duplicates are observed, the e-cash system has a designed method to subsequently determine the identity of the offender who performed the double spending. Yet, quite a few schemes have been devised to combat the double spending problem while achieving various other objectives, such as the *offline payment*, *untraceability* and *transferability*.

The techniques of defense against double spending in e-cash can be used in mobile agent clone detection systems. In both cases, illegal duplications of files are investigated. More of their analogies are discussed later. To a better understanding of our mobile agent clone detection system, we introduce two e-cash systems in this chapter.

The first one is Chaum-Pedersen's general transferable e-cash system [ChePe93]. This scheme adds transferability to a large class of the existing offline untraceable e-cash systems. We transplant this scheme to our general mobile agent clone detection system in chapter 5.

The second one is Ferguson's single term offline e-coin system [Fer93]. His scheme is one

of the efficient e-cash systems that satisfy Chaum-Pedersen's general e-cash model. Wong shows that we can add transferability to Ferguson's scheme with Chaum-Pedersen's scheme [Won01]. We use this transferable Ferguson's e-coin to demonstrate the specific implementation of our mobile agent clone detection system in chapter 6.

4.2 The General E-cash Model

This section presents a general e-cash model proposed by Chaum-Pedersen [ChaPe93]. The model is generic in the sense that it applies for a large class of *offline* electronic payment systems, which is composed of three protocols: the Withdrawal Protocol, the Payment Protocol and the Deposit Protocol. For simplicity, we assume that the payer always spends the total value of the coin and no refund of unused value is considered.

The bank is in possession of a secret key S_I and the corresponding public key P_I. The signature on the message m by the secret key S_I, denoted $S_I(m)$, is worth a fixed amount of money (say \$1). The signature is issued blindly, i.e., the bank issues the signature $S_I(m)$ to the user, without getting any information from m. Payment by the user is made *offline*, i.e. only the payer and the payee are involved, without any supervision from the bank. The payee can deposit the coin to the bank at any time. The offender who performed the *double spending* could be identified from the joint content of the duplicated coins.

The general e-cash model is sketched below:

Withdrawal

1. User U_1 creates a message m_1 of a special form (see later) with the bank, without giving the bank any information of m_1. U_1 verifies the correctness of m_1.
2. The bank sends the blind signature of m_1 to U_1 and withdraws \$1 from U_1's account.
3. U_1 unblinds the signature to yield $S_I(m_1)$.

Payment

1. U_1 pays another user U_2 by sending her m_1 and $S_I(m_1)$.
2. U_2 verifies $S_I(m_1)$ is the signature of m_1 with P_I. Then U_2 chooses a challenge c_1 and sends it to U_1.
3. U_1 constructs a response r_1 to c_1 in a special form (see later) and sends it to U_2.
4. U_1 verifies that r_1 is correct corresponding to c_1 and m_1.

Remark: In order to investigate the double-spender, m_1 and r_1 are formed in special forms: No

two different coins issued are with the same m_1, i.e. each coin has unique m_1. If U_1 gives correct responses to two different challenges, the bank can identify that U_1's is the double spender. In contrast, a single correct response gives no information of U_1's identity. The disability in tracing the identity of honest spender is called *untraceability*.

Deposit

1. U_2 deposits the coin to get \$1 by sending m_1, $S_1(m_1)$, c_1 and r_1 to the bank.
2. The bank verifies that $S_1(m_1)$ is the signature of m_1. Then the bank verifies that r_1 is the correct response to c_1 and m_1. Then the bank increases U_2's account with \$1. The coin is stored in the checklist for future checking of double spending.

The bank checks whether m_1 is deposited previously from its checklist. In the case of positive incidence, the double spender identity U_1 is revealed from the two challenge-and-response pairs, provided that the two challenges are distinct.

The above general e-cash model outlined the fundamental elements for an e-cash system. However, transferability is not addressed in the above model, where a user cannot further spend the coin she just received without depositing it to the bank. In [ChaPe93], transferability is added on top of this general model.

4.3 Chaum-Pedersen's General Transferable E-cash

This section introduces Chaum-Pedersen's general transferable e-cash, based on the general e-cash model described in the previous section. The major modifications are made in the payment protocol, where no contact to the bank is required between two payments. To add transferability, a *zero-value* coin with signature $S_0(m)$ and a one-way function f is required.

Besides the key pair (S_1, P_1), another key pair (S_0, P_0) is needed by the bank in this scheme. The signature of m by S_0, denoted $S_0(m)$, has the same properties as $S_1(m)$ except that $S_0(m)$ worths nothing. Prior to the payment, a user withdraws from the bank a number of distinct zero-value coins signed with S_0. Unlike the withdrawal protocol of a valued coin, no money is taken out from the user's account.

Furthermore, instead of generating the challenge at a truly random manner, the challenge c is constructed as

$$c = f(m, \rho)$$

where m comes from a zero-value coin and ρ is formed in a special way, which ensures that the

payee can later deposit the money if she wants and ensures that the payer receives different challenges, if she tries to spend the same coin twice, even the payer colludes with the payee.

The payment protocol for a transferable coin is described below:

Payment of a transferable coin (The ℓ'th payment)

1. The payer U_ℓ sends m_1, $S_1(m_1)$, r_i, m_{i+1}, $S_0(m_{i+1})$ and ρ_{i+1} to the payee $U_{\ell+1}$, for $1 \le i \le \ell - 1$.
   (For $\ell = 1$, only m_1 and $S_1(m_1)$ are sent)
2. $U_{\ell+1}$ verifies all the signatures with P_0 and P_1.
   $U_{\ell+1}$ verifies r_i is a correct response to the challenge $f(m_{i+1}, \rho_{i+1})$, for $1 \le i \le \ell - 1$.
   (For $\ell = 1$, the response verification is omitted.)
   $U_{\ell+1}$ computes its challenge $c_\ell = f(m_{\ell+1}, \rho_{\ell+1})$ and sends it to U_ℓ.
3. U_ℓ constructs a response r_ℓ to c_ℓ in a special form and sends it to $U_{\ell+1}$.
4. $U_{\ell+1}$ verifies r_ℓ is correct corresponding to c_ℓ and m_ℓ.

Remark: ℓ is implicitly computed from the number of zero-value coins received for payment.

Suppose $U_{\ell+1}$ deposits the coin to the bank after ℓ'th payment. If the bank finds from the checklist that m_1 has already deposited before, then each (c_i, r_i) pair is investigated, where $c_i = f(m_{i+1}, \rho_{i+1})$. The double spender identity U_i is revealed from two distinct (c_i, r_i) pairs from the coin deposited and the coin in the checklist.

It is easy to implement the above scheme to a known e-cash system fitting the general model. The untraceability of this transferable coin is inherited from the specific e-cash system used for implementation. However, in Chaum-Pedersen's scheme, a payer can always recognize her coin when she sees the coin again in later payment. Furthermore, the coin grows in size after each payment. Later we will show that the above scheme could be transplanted to construct a mobile agent clone detection system.

4.4 Ferguson's Single-term Off-line E-coins

Ferguson proposed a single-term offline untraceable e-cash system [Fer93]. The system involves three parties: the Bank, the User and the Shop. A User withdraws a coin from the Bank. Then the User can pay this coin to the Shop. The Shop finally deposits the coin to the Bank. Ferguson's scheme is an offline scheme as its payment involves only the two parties in the transaction, but not the Bank or any other third party.

With certain cryptographic techniques, we cannot trace the payment history from the content of the coin. However, if the coin is spent twice (or more), the Bank can identify the person who double spent the coin and prosecute the double spender(s).

As we can see, Ferguson's scheme satisfies the requirements of Chaum-Pedersen's general e-cash model. Wong adds transferability to Ferguson's scheme by using Chaum-Pedersen's techniques. We demonstrate the specific implementation of our general mobile agent clone detection system by this transferable Ferguson's e-coin in a later chapter.

In this section, we first introduce some technical backgrounds as a basis for the security in Ferguson's scheme. Then we present the details of each protocol in his scheme.

4.4.1 Technical Background of the Secure Tools

A number of cryptographic tools are employed in Ferguson's coin, including the Secure Hash Function, the Polynomial Secret Sharing Scheme and the RSA-based Randomized Blind Signature. The security of Ferguson's scheme is based on the security of these cryptographic tools. Details are described as follow.

Secure Hash Function. The one-way and collision resistant hash function is an important cryptographic primitive in e-cash systems. Given an one-way and collision resistant hash function f and a value y, it is computationally infeasible to find a value x, such that $y = f(x)$. Also, it is computationally infeasible to find $x_1 \neq x_2$, such that $f(x_1) = f(x_2)$. Secure hash function is used in Ferguson's e-coin and its security is based on the above properties.

Polynomial Secret Sharing Scheme. One of the objectives in the design of e-cash systems is to detect double-spending of coins. For offline schemes such as Ferguson's e-coin, double-spending cannot be detected immediately during the payment. Instead, the fraud is detected 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 clues to the user's identity. But two such releases are sufficient to identify the user uniquely. The secret sharing scheme (refer to section 2.6 for details) is employed for this purpose.

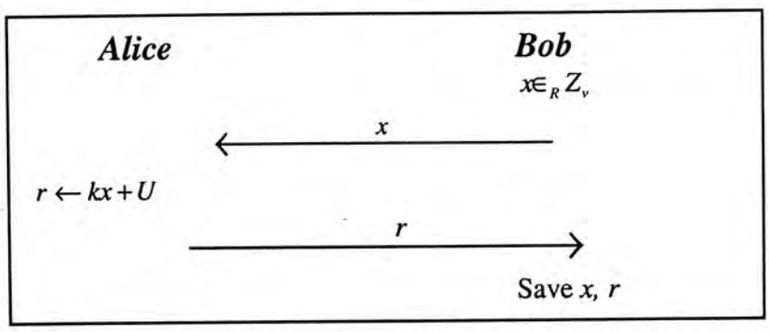


Figure 4.1: Shamir's Polynomial Secret Sharing Scheme

Shamir's polynomial secret sharing scheme [Sha97] is hired by Ferguson for the double-spending detection. The central idea is based on the polynomial interpolation, where a univariate polynomial $y = f(x)$ of degree $t - 1$ is uniquely defined by t points (x_i, y_i) with distinct x_i. We can do it by solving t linearly independent equations with t unknowns.

In the double spending detection, two distinct pairs of challenges and responses are needed to identify the double spender, as shown in Figure 4.1.

Alice has her identity U and a secret number k.

1. Bob randomly generates a challenge x and sends it to Alice.
2. Alice responses $r = kx + U$ to Bob.
3. Bob saves x and r for later detection.

As a result, one (x, r) pair cannot reveal Alice's identity while two such pairs can identify Alice by solving the two equations with two variables, provided that the two challenges are different.

RSA-based Randomized Blind Signature. Privacy is one of the designed objectives of an e-cash system. To prevent the Bank and the Shop from extracting the User's (Alice's) identity U from the coin, Ferguson employs the RSA-based Randomized Blind Signature Scheme proposed by Chaum [Cha92]. This signature scheme satisfies the following properties:

- *Alice receives an RSA-signature on a number of a special form, which she cannot create herself*
- *The Bank is sure that the number it signs was randomly chosen.*
- *The Bank receives no information regarding which signature Alice gets.*

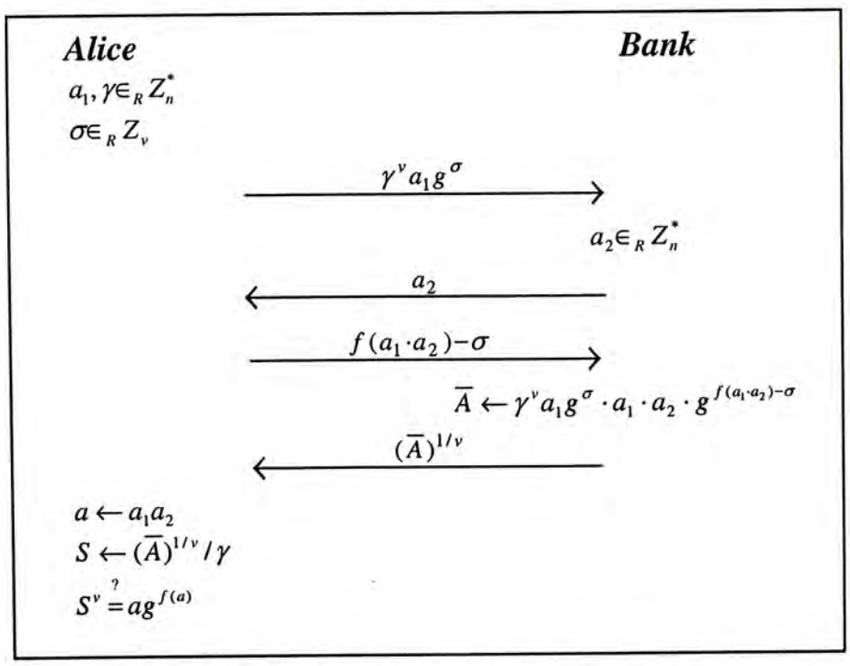


Figure 4.2: RSA-base Randomized Blind Signature

The Bank publishes its RSA public key pair (v, n), a one-way function f mapping from Z_n^* to Z_v, and an integer g of large order in Z_n^*.

Step 1 Alice chooses a random number $a_1 \in Z_n^*$, a multiplicative blind factor $\sigma \in_R Z_n^*$, and an exponential blind factor $\gamma \in_R Z_v$. Alice computes $\gamma^v a_1 g^\sigma$ and sends it to the Bank.

Step 2 The Bank chooses its own contribution a_2 and sends it back to Alice.

Step 3 Alice computes $f(a_1 \cdot a_2) - \sigma$, and sends its back to the 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 $\gamma^v a_1 a_2 g^{f(a_1 \cdot a_2)}$. The Bank computes the v'th root of this number and sends it (the blind signature) to Alice.

Step 5 Alice unblinds the blind signature by dividing it by γ to yield the pair $(a, (ag^{f(a)})^{1/v})$, where $a = a_1 a_2$ is called the base number of the signature.

The above signature scheme satisfies the following requirements:

1. **Alice cannot create the signature herself:** *It is computationally infeasible to forge a signature pair of the form* $(a, (ag^{f(a)})^{1/v})$. To forge the signature pair (a, A), we try to solve $A^v = ag^{f(a)}$. The first way is do it to fix A and test for different a by trial and error. The probability of success is negligible, which is equal to $1/v$. The second way is to fix a, and compute A from $(ag^{f(a)})^{1/v}$. However, this method requires computation of the v'th root, which is as difficult as breaking the RSA-key algorithm. No general methods are currently known that attempt to break RSA in this way. The third way is to use the existing signature pairs to create a new signature pair. Nevertheless, the properties of the one-way function f restrict Alice from getting a valid signature by any combinations of these existing signatures.
2. **The numbers signed by the Bank are randomly chosen:** The base number a is derived from $f(a_1 \cdot a_2)$, where a_1 is the contribution from Alice, while a_2 is contributed by the Bank. This method uses two random numbers, which ensures that the message signed by the bank is randomly chosen.
3. **The Bank receives no information regarding which signature Alice gets:** For the Bank to recognize Alice from the coin signature, she should find the values of a or $ag^{f(a)}$. From the communications between the Bank and Alice, the Bank only knows a_2, while a_1 and $f(a)$ are hidden by the randomly generated blind factors γ and σ. From the point of view of the Bank, all possible signature pairs are equally likely.

The above cryptographic tools are essential for the security of Ferguson's e-cash system. In the next section, we show how Ferguson makes use of these tools to model an efficient single term offline untraceable e-cash system.

4.4.2 Protocol Details

Ferguson's e-cash system consists of three protocols: the Withdrawal Protocol, the Payment Protocol and the Deposit Protocol. This system needs three base numbers a, b, and c, one secret number k, the user identity U, and two RSA signatures S and T from the bank. A valid e-coin must satisfy the following equations:

$$A = ag_1^{f(a)}, B = bg_2^{f(h_b^b)}, C = cg_3^{f(h_c^c)}$$

$$S = (C^k A)^{1/v}, T = (C^U B)^{1/v}$$

where the RSA public key v and the RSA modulus n of the Bank are known to the public. The parameter v is assumed to be a large prime. The private key $1/v$ of the Bank is kept secret in the Bank. The parameters g_1, g_2, g_3 are publicly known elements of large orders in the multiplicative group of Z_n^*. The parameters h_3 an h_4 are publicly known elements of orders n in the multiplicative group Z_p^*, where the parameter p is a large prime where p-1 is a multiple of n. The mapping f is a suitable one-way function from Z_n^* to Z_v^*.

Throughout this paper, we assume computations are done modulo n while those involve exponent are done modulo v unless otherwise specified.

With the RSA-based randomized blind signatures, a user (Alice) cannot forge a coin with a valid signature without knowing the Bank's private key $1/v$, while the Bank cannot get any information about Alice from the coin. With the polynomial secret sharing scheme, no one can trace the identity of honest parties that involved in the transaction from the content of the coin, while the identity of the offender who performed double spending could be extracted from the coin's content on deposit.

The details of each protocol are described as below:

Withdrawal Protocol: The withdrawal protocol consists of three parallel runs of RSA-based randomized blind signature. Two of the runs are the restricted version, and one is the unrestricted version. It is assumed that Alice authenticated her identity U with the Bank prior to the protocol.

Step 1: Alice prepares three random numbers as her contributions to the base numbers: a_1, b_1, $c_1 \in_R Z_n^*$. Then she prepares three random numbers as the multiplicative blinding factors: $\gamma, \alpha, \beta \in_R Z_n^*$ and three random numbers as the exponential blinding factors: $\sigma, \tau, \phi \in_R Z_v$. Then Alice computes $\gamma^v c_1 g_3^\sigma, \alpha^v a_1 g_1^\tau$, $\beta^v b_1 g_2^\phi$ and sends

them to the Bank.

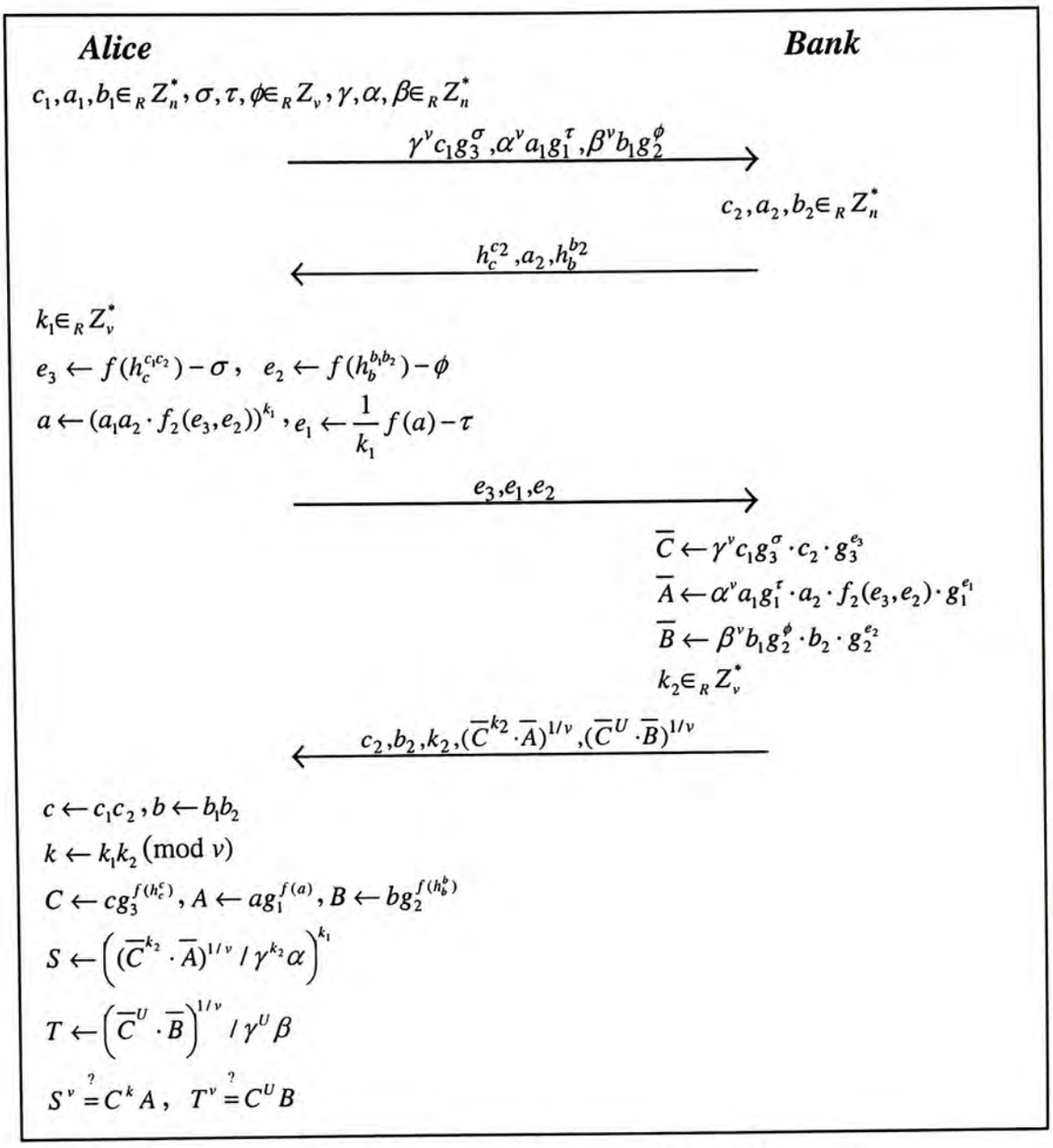


Figure 4.3: The withdrawal protocol of Ferguson's e-cash system

Step 2: The Bank computes her three contributions to the base numbers at random: a_2, b_2, $c_2 \in_R Z_n^*$. Then she sends $h_c^{c_2}, h_b^{b2}$ and a_2 to Alice. Sending a_2 directly allows Alice to raise one of the resulting signatures to a power she chooses.

Step 3: Alice chooses a random number $k_1 \in_R Z_v^*$ and computes $e_3 \leftarrow f(h_c^{c_1 c_2}) - \sigma$, $e_2 \leftarrow f(h_b^{b_1 b_2}) - \phi$, $a \leftarrow (a_1 a_2 \cdot f_2(e_2, e_3))^{k_1}$ and $e_1 \leftarrow \frac{1}{k_1} f(a) - \tau$, where f_2 is a publicly known one-way function mapping from $Z_n^* \times Z_n^*$ to Z_v^*. Alice sends e_1, e_2 and e_3 to the Bank.

Step 4: The Bank computes the blind version of A, B and C, with $\overline{C} \leftarrow \gamma^v c_1 g_3^\sigma \cdot c_2 \cdot g_3^{e_3}$,

$\overline{B} \leftarrow \beta^v b_1 g_2^{\phi} \cdot b_2 \cdot g_2^{e_2}$ and $\overline{A} \leftarrow \alpha^v a_1 g_1^{\tau} \cdot a_2 \cdot f_2(e_2, e_3) \cdot g_1^{e_1} = \alpha^v a g_1^{(1/k_1) f(a)}$.

The blinded values and the unblinded values satisfy $\overline{C} = \gamma^v C$, $\overline{A} = \alpha^v A^{1/k_1}$ and $\overline{B} = \beta^v B$. The Bank randomly chooses a number $k_2 \in_R Z_v^*$ and sends c_2, b_2, k_2, $(\overline{C}^{k_2} \cdot \overline{A})^{1/v}$ and $(\overline{C}^U \cdot \overline{B})^{1/v}$ to Alice. Alice's account was decreased by the value specified by the coin withdrawn.

Step 5: Alice computes $b = b_1 b_2$, $c = c_1 c_2$ and $k = k_1 k_2$ and constructs the numbers $C \leftarrow c g_3^{f(h_c^c)}$, $A \leftarrow a g_1^{f(a)}$ and $B \leftarrow b g_2^{f(h_b^b)}$. Alice computes the two signatures $S \leftarrow \left((\overline{C}^{k_2} \cdot \overline{A})^{1/v} / \gamma^{k_2} \alpha \right)^{k_1}$ and $T \leftarrow (\overline{C}^U \cdot \overline{B})^{1/v} / \gamma^U \beta$. Alice verifies the signatures are correct if $S^v = C^k A$ and $T = C^U B$.

Alice finally obtains the base numbers *a, b, c,* the secret *k,* and the signatures *S, T,* which depends on $1/v$ at the end of the protocol. These six numbers together with Alice identity *U* are used as inputs to the payment protocol.

Payment Protocol: Polynomial secret sharing scheme is employed in the payment protocol. Partial information of Alice's identity is embedded into the coin. The Shop knows nothing about Alice's identity from the received coin, but is sure that the coin is belong to Alice, by a single challenge and response.

Step 1: Alice sends the base numbers *a, b, c* to the Shop

Step 2: The Shop replies Alice with a random challenge $x \in_R Z_v^*$.

Step 3: Alice responds with $r = kx + U$ and $R = S^x T$.

Step 4: The Shop computes *A, B, C* from *a, b, c.* Then she verifies that $R^v = C^r A^x B$ and accepts the payment.

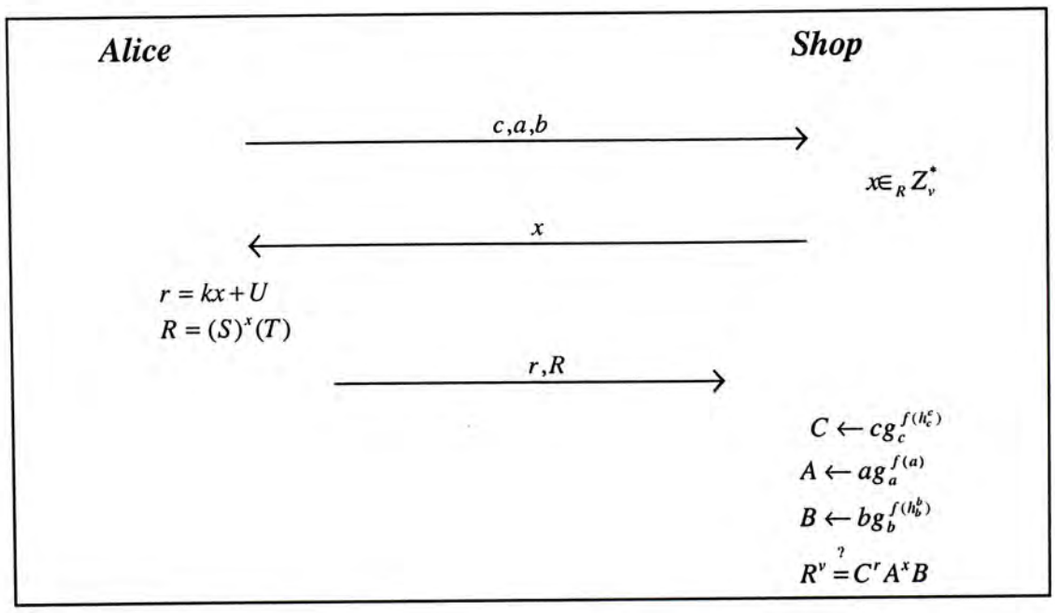


Figure 4.4: The payment protocol of Ferguson's e-cash system

Since Alice cannot create the signature herself, she cannot change the values of k and U in the challenge and response. This ensures that the true identity of the offender who performed the double spending will be correctly figured out.

Deposit Protocol: To increase the balance of the Shop's account, the Shop deposits the received coin to the Bank.

Step 1: The Shops sends the base numbers *a, b, c,* the challenge x ,and the response r and R to the Bank.

Step 2: The Bank computes *A, B, C* from *a, b, c* and verifies that $R^v = C^r A^x B$. If it passes the verification, the Bank increases the Shop's account with the amount specified by the coin.

If Alice spends the coin twice, she must reveal two different points on the secret sharing line $r \leftarrow kx + U$, which immediately allow the Bank to determine her identity.

Ferguson's e-cash system is offline and untraceable. However, it is not transferable, i.e. a user should deposit the received coin to the bank before she can spend it again. Wong adds transferability to Ferguson's e-coin by Chaum-Pedersen's method. This transferable Ferguson's e-coin will be used for the specific implementation of our general mobile agent clone detection system.

4.5 Conclusion Remarks

The e-cash and the mobile agent are two quite different fields of study. However, they have similarities in their security goals. The e-cash system requires a mechanism to restrict users from duplicating and reusing the e-cash. The mobile agent system prohibits any illegal copies of agent from multiple executions in the distributed computing environment. In both cases, illegitimate duplication of the electronic file is detected and the offender who made it is identified.

Motivated by this observation, we try to transplant the double spending detection scheme to the mobile agent clone detection system. By using Chaum-Pedersen's general transferable e-cash model, we can apply the system to a large class of the existing e-cash schemes. One of the examples is Ferguson's offline untraceable e-cash. Wong modified his scheme with Chaum-Pedersen's techniques to add transferability to the e-cash system. We will show that this scheme can be further extended to implement our mobile agent clone detection system.

Chapter 5

A Mobile Agent Clone Detection System using General Transferable E-cash

5.1 Introduction

The mobile agent technology offers a new computing paradigm in which an autonomous program, working on behalf of its owner, can suspend its execution on a host computer, migrate itself to another agent-enabled host on the network, and resume the execution on the new host, according to [JK99]. Thanks to its autonomy and mobility, the concept of a mobile agent is applied to areas such as workflow systems, electronic commerce and information retrieval systems. However, security threats are still the deployment bottleneck.

A number of surveys on these security threats and its counter-measures are studied, ranging from risking a host by a malicious agent, to risking an agent by a malicious host [FGS96, Che98, JK99, NC99, GM00, Jan99]. These surveys addressed that unauthorized mobile agent cloning is an open issue that is difficult to solve. Baek [Bae98] proposed a clone detection system in which a central site oversees every agent migration.

In this chapter, we propose a new scheme for the detection of the unauthorized cloning of mobile agents using the general transferable e-cash proposed by Chaum-Pedersen [ChaPe93]. By using a suitable e-cash system, we can implement a mobile agent clone detection system with the culprit host(s) which performed the cloning being uniquely identified. Our clone detection scheme cannot prevent the cloning from happening. Rather, it identifies the offending host after cloning occurs and so that the culprit may be penalized. Bredin, et al. [BKR98] also proposed

an e-cash based system for mobile agent security.

Besides investigation of the offending host, other benefits are added into our scheme. One is the *itinerary privacy*, where the identity of the honest host traversed by the agent cannot be determined from the information carried by the agent. Another one is *offline computation*, where an agent's migration involves only local computations by the agent, the sending host and the receiving host. No central oversight from exterior party such as those in Baek's [Bae98] scheme is required.

Our scheme is based on a cryptographic technique called *e-cash*. An *e-cash* is the electronic counterpart to the physical cash (paper bills and coins) in real life. It can be used in electronic transactions as a medium of payment. In a *transferable e-cash*, the payment recipient can immediately use the received e-cash to pay another entity.

An e-cash is ultimately an electronic file. Therefore, it is easy to produce an exact duplicate of an e-cash. One of the security requirements in the design of an e-cash system is that any duplicate of an e-cash cannot be spent again. The problem of e-cash duplication and its multiple uses is called the *double spending* problem. E-cash is an active area of research in cryptography [CFN88, vA90, OO91, Fer93, Bra93, Won01]. Quite a few schemes have been devised to combat the double spending problem while achieving various other objectives. In many schemes, double spending cannot be prevented from happening. Rather, the e-cash system examines the circulations in the system and checks for duplicates. When duplicates are observed, the e-cash system has a designed method to subsequently determine the identity of the offender who performed the double spending.

One of our central motivations is the observation that double spending in e-cash has several similarities to the problem of unauthorized cloning of mobile agents. Mobile agents are also ultimately electronic files, and thus can be exactly duplicated with ease. In order to devise a mobile agent clone detection system, one can look for existing techniques from e-cash systems.

Our scheme for detecting the unauthorized cloning of mobile agents is motivated by Chaum-Pedersen's general transferable e-cash [ChaPe93]. With Chaum-Pedersen's general transferable e-cash model, we can implement a mobile agent clone detection system with a large class of the existing e-cash schemes. By modifying and extending some key techniques, we are able to design a system which can determine whether two given agents are clones of each other. In the case that two clones are observed, our system can determine, from the information contained in the clones, the identity of the host who performed the cloning. The security and privacy of our scheme depends on the particular e-cash scheme used for implementation.

The rest of the chapter is organized as follows: Section 5.2 briefly explains some terminologies required for understanding our mobile agent clone detection system. Section 5.3 gives details of our system based on Chaum-Pedersen's general transferable e-cash model.

Security and privacy of our system are analyzed in Section 5.4. Then we elaborate the security and the privacy issues with two attack scenarios in Section 5.5. Section 5.6 introduces a simple alternative scheme for offline detection of mobile agent, where itinerary privacy is not a concern. Finally we have a conclusion in Section 5.7.

5.2 Terminologies

In this section, we introduce a few terminologies used in our mobile agent clone detection system. They include *mobile agent, authorized clone, unauthorized clone, offline migration,* and *itinerary privacy*. These terminologies will be repeatedly mentioned in future discussions.

Mobile Agent: A mobile agent in our scheme is composed of a unique identifier I, a code logic $\mathcal{K}$ and a state $\mathcal{S}$. I and $\mathcal{K}$are unaltered, while $\mathcal{S}$ could be updated after each migration from host to host. We denote an agent as (I, $\mathcal{K}$, $\mathcal{S}$) in this chapter.

Authorized Clone: Some mobile agent systems, such as Aglet system [KLO97], support the agent clone generation facility. Some literatures even use this clone property for the purpose of load balancing in distributed systems [She99, SSCJ98, Fan01]. Nevertheless, the clone in these systems has a different identifier from its original agent, although it has the same code and state as the original one at a given time. This class of cloning, with different identifiers, is regarded as an *authorized clone*. The authorized clones could be distinguished from their identifiers.

Unauthorized Clone: Baek [Bae98] redefines a clone as the copied agent, which has the same the identifier as the original one. Based on his definition, it is easy to make a clone without any knowledge on mobile agent systems, such as Aglet and Mole [KLO97, SBH96], causing security threats. This class of cloning, with the same identifier, is regarded as an *unauthorized clone*. We cannot distinguish between two clones from each other. In this thesis, we focus on the unauthorized cloning of mobile agent.

Offline Migration: The migration of a mobile agent form one host to another is offline if it requires no supervision from any exterior party, other than the sending host, the receiving host and the agent.

Itinerary Privacy: The itinerary of a mobile agent is the record of hosts where the agent has ever visited. With itinerary privacy, the identity of any honest host could not be traced using the information carried by the agent.

After understanding the terminologies, we can proceed to discussing the protocols of our mobile agent clone detection system. As you will see, our system is focused on the

unauthorized cloning of mobile agent, with the advantages of offline migration and itinerary privacy.

5.3 A Mobile Agent Clone Detection with Transferable E-cash

This section presents a clone detection system for unauthorized cloning of mobile agent using Chaum-Pedersen's general transferable e-cash (refer to chapter 5 for details). In both cases, illegitimate duplicates of files (e-cash/mobile agent) are investigated and the offender is identified after the incidence. If the e-cash system used is untraceable, then the mobile agent clone detection system benefits from itinerary privacy.

In Chaum-Pedersen's general transferable e-cash, two sets of coins are used. They are the valued coin signed with the bank's secret S_1 and the zero-value coin signed with the bank's secret S_0. The main idea of our scheme is to bind certain important information of an agent into a valued coin during the withdrawal. The agent (I, $\mathcal{K}$, S) carries this coin from one host to another (payment). The host adds a zero-value coin to the visited agent, with its identity embedded in the coin. Correct binding of the mobile agent to the coin is verified during each migration with the bank's public keys P_0 and P_1. If duplicates of a coin are found, with a correct binding to the mobile agents, then the mobile agents are clones. The offender could be identified using similar techniques dealing with the double spending of e-coins.

To summarize, the protocols are built on top of Chaum-Pedersen's general transferable e-cash (refer to chapter 4 for details), with the following major modifications:

1. We are not concerned about the values of the coins signed with S_0 and S_1, although we still call them the valued coin and the zero-value coin respectively.
2. The identifier I and the code $\mathcal{K}$ of a mobile agent are used as inputs for signature of a valued coin. Each agent with a unique I could obtain one valued coin only.
3. Payment of a coin is now replaced by sending an agent carrying the coin. Correct binding of the mobile agent to the coin is verified for each migration/payment.
4. Deposit of a coin is not required. Instead, culprit identification is needed for investigation of offenders who performed unauthorized cloning of agents.

For simplicity, we assume that no agent is allowed to revisit any host that it has traversed before. Following are the details of the protocols:

Withdrawal of a valued coin

1. Host U_1 creates an agent $(I, \mathcal{K}, \mathcal{S})$ and sends I and $\mathcal{K}$ to the bank.
   Host U_1 creates a message m_1 of a special form (see later) with the bank, without giving the bank any information of m_1. U_1 verifies the correctness of m_1.
2. The bank checks whether it is the first withdrawal for the agent with identifier I. If yes, the bank sends the blind signature of m_1, I and $\mathcal{K}$ to U_1.
3. U_1 unblinds the signature to yield $S_1(m_1, I, \mathcal{K})$.

Withdrawal of a zero-value coin

1. Host U_i creates a message m_i of a special form (see later) with the bank, without giving the bank any information of m_i. U_i verifies the correctness of m_i.
2. The bank sends the blind signature of m_i to U_i.
3. U_i unblinds the signature to yield $S_0(m_i)$.

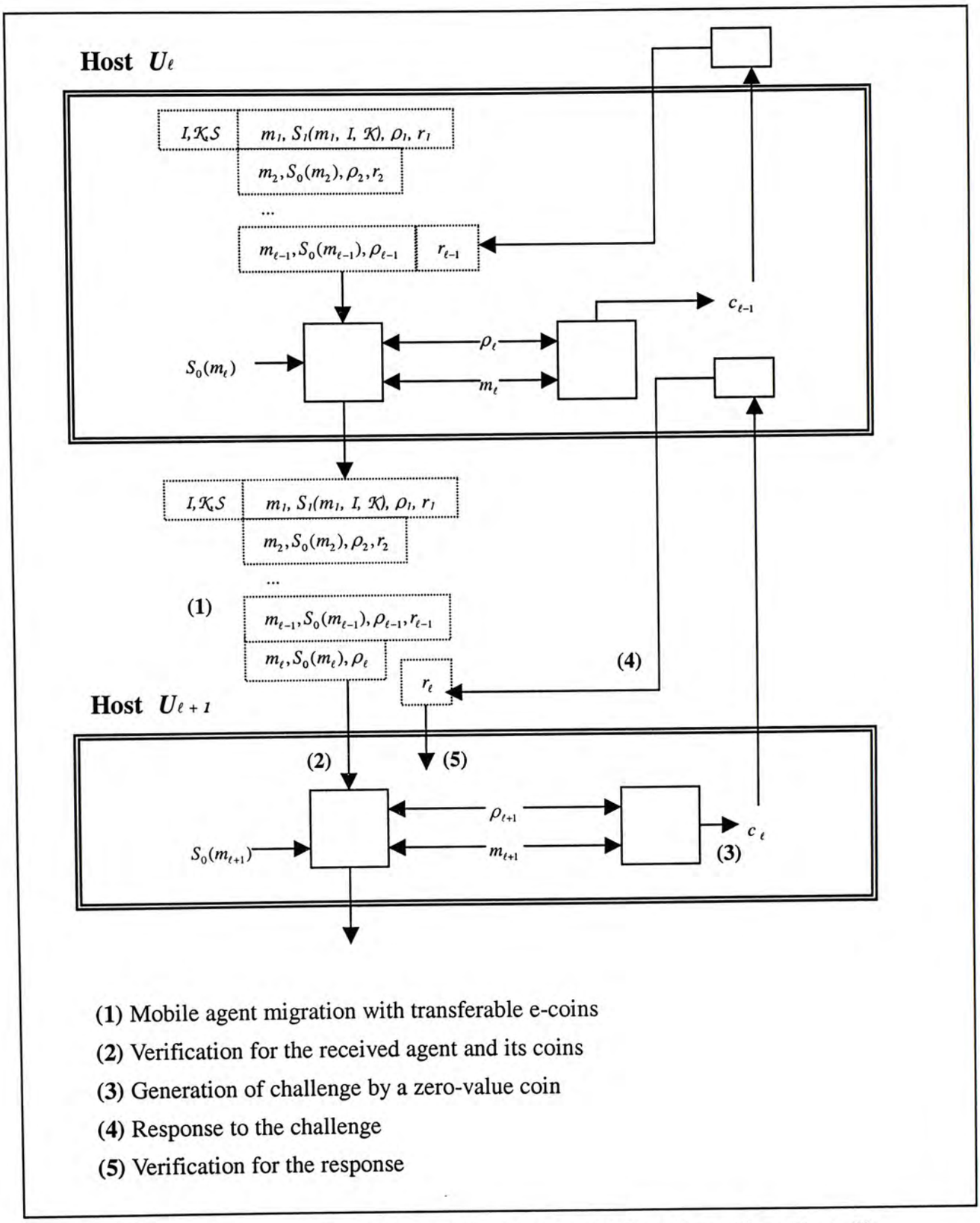
Remark: To receive and send-away an agent, a host should withdraw a number of distinct zero-value coins beforehand.

Migration (The ℓ'th migration)

1. The sending host U_ℓ sends $(I, \mathcal{K}, \mathcal{S})$, m_1, $S_1(m_1, I, \mathcal{K})$, r_i, m_{i+1}, $S_0(m_{i+1})$ and ρ_{i+1} to the receiving host $U_{\ell+1}$, for $1 \le i \le \ell - 1$. (For $\ell = 1$, only $(I, \mathcal{K}, \mathcal{S})$, m_1 and $S_1(m_1)$ are sent)
2. $U_{\ell+1}$ verifies all the signatures with P_0 and P_1.
   $U_{\ell+1}$ verifies r_i is a correct response to the challenge $f(m_{i+1}, \rho_{i+1})$, for $1 \le i \le \ell - 1$.
   (For $\ell = 1$, the response verification is omitted.)
   $U_{\ell+1}$ computes its challenge $c_\ell = f(m_{\ell+1}, \rho_{\ell+1})$ with an unused zero-value coin $m_{\ell+1}$ and sends it to U_ℓ.
3. U_ℓ constructs a response r_ℓ to c_ℓ in a special form (see later) and sends it to $U_{\ell+1}$.
4. $U_{\ell+1}$ verifies r_ℓ is correct corresponding to c_ℓ and m_ℓ.

Remark: ℓ is implicitly computed from the number of the zero-value coins received for payment.

Remark: In order to investigate the offender who cloned the agent, m_i and r_i are formed in special forms: No two different coins issued are with the same m_i, i.e. each coin has unique m_i. If U_i gives correct responses to two different challenges, the bank can identify that U_i is the cloning host. In contrast, a single correct response gives no information of U_1's identity. The disability in tracing the identity of an honest host is called *itinerary privacy*. Note that the above migration protocol is offline, i.e., no centralized oversight from the bank is required.



(1) Mobile agent migration with transferable e-coins

(2) Verification for the received agent and its coins

(3) Generation of challenge by a zero-value coin

(4) Response to the challenge

(5) Verification for the response

Figure 5.1: The $\ell'th$ migration of the mobile agent $(I, \mathcal{K}, S)$ from Host U_ℓ to Host $U_{\ell+1}$

Culprit Identification

Two agents (I, $\mathcal{K}$, S) and (I', $\mathcal{K}'$, S') are unauthorized clones if and only if $I = I'$ and $\mathcal{K} = \mathcal{K}'$. Any party can independently determine whether two given agents, whether visiting its sites or being transmitted from elsewhere, are clones.

The culprit identification is based on the fact that two agents (carrying valid coins and responses) are unauthorized clones if and only if they carry the same valued coin m_1 with the signature $S_I(m_1, I, \mathcal{K})$. There are two reasons for this fact to be true. First, in the general e-cash model, it requires that each coin has unique m_1. Second, in our scheme, the bank only issues one valued coin to each unique agent.

Remark: An agent carrying invalid coins and responses will fail in the verifications of migration protocol and is not able to travel onward. We consider them less harmful and not investigate them in this thesis.

From the information carried by the two agent clones, we have

$$m_1, m_1',$$
$$r_i, m_{i+1}, \rho_{i+1} \quad \text{for } 1 \le i \le \ell - 1$$
$$\text{and} \quad r_i', m_{i+1}', \rho_{i+1}' \quad \text{for } 1 \le i \le \ell' - 1.$$

Then we compute

$$c_i = f(m_{i+1}, \rho_{i+1}) \text{ for } 1 \le i \le \ell - 1$$
$$\text{and} \quad c_i' = f(m_{i+1}', \rho_{i+1}') \text{ for } 1 \le i \le \ell' - 1$$

Suppose i is the smallest integer such that $m_i \neq m_i'$. We first focus on the case $i > 1$. From the special constructions of m and r in the general e-cash model, the identity of the host which made the clones could be uniquely identified by two challenge and response pairs (c_{i-1}, r_{i-1}) and (c_{i-1}', r_{i-1}').

If i=1, then these two clones are still in the possession of their original agent-creating host and have never migrated yet.

If i does not exist and $\ell = \ell'$, then the agent is cloned, however, the cloning host has not yet passed either of them out. If i does not exist and $\ell < \ell'$, then the cloning host only passed only one of clones out. These two cases are not considered as crime.

5.4 Security and Privacy Analysis

This section discusses the security and the privacy aspects of our system. The security and the privacy of our scheme is mainly based on:

1. How the e-cash system used for the implementation matches the requirements of the general e-cash model.

2. How the one-way function f and the special element ρ are implemented.

Following are some key requirements that the e-cash system should meet, or at least closely meet, in order to ensure the security of our scheme.

Blind signature

1. The bank should give the signature on m by S_i without getting any information of m from the signature, where $i = 0$ or 1. Otherwise, the coin is traceable by the bank and the itinerary privacy is lost.
2. The signature could not be forged by the host without the knowledge of the corresponding secret S_i, where $i = 0$ or 1. Otherwise, the host could embed a fake identity to the coin herself, which makes errors in the culprit identification.

See [Cha92] for examples of such signature scheme. Forging of signatures in [Cha92] is as hard as breaking RSA-signature [RSA78] or trails and errors with negligible probability of success.

Constructions and correct uses of m and r

1. m is unique for each coin. Otherwise, we cannot distinguish whether two coins carried by the agents come from the same copy or from two distinct coins with the same m.
2. Every unique agent (I, $\mathcal{K}$, S) can withdraw a unique valued coin with m_I and $S_I(m_I, I, \mathcal{K})$ only. Otherwise, if two agent clones are sent immediately after it is created, with $m_I \neq m_I'$, we cannot compute the challenges and responses, and the subsequent investigation of the offender will fail.
3. The coin m with the signature $S_0(m)$ cannot be reused for multiple migrations of agents. Otherwise, it is vulnerable to the *chosen host response attack*, where an honest host is traced due to the reuse of a coin for different migrations. Details are discussed later.
4. The response r constructed by m should be in such a way that a single response gives no information about the host's identity, but two such responses to different challenges enable the identification of the host which gives the responses. Otherwise, either the honest host is traceable, or the offending host can escape from prosecution.

See [Sha79] as an example for such conditional reveal of information.

The one-way function f and the special element ρ

1. The special element ρ has the property that we cannot find two different (m, ρ) pairs which generates the same challenge $c = f(m, \rho)$. Otherwise, a host may replace m by another coin with m', where $c = f(m', \rho')$, passing the verifications in migration protocol, when she sends away the agent. When the two agent clones are compared, since one

response is constructed by m, while the other one is constructed by m', the identification of the cloning host fails.

One example for the implementation of the above property is by the one-way collision resistant hash function, such as SHA-1. First, the hash function gives randomness to the challenge. Second, as it is collision resistant, it is computationally infeasible to find two different inputs yielding the same output. Refer to [MOV97] for details.

As Chaum-Pedersen's general transferable e-cash, our mobile agent clone detection system is described in a general aspect. The security and the privacy depend on the specific e-cash system used for the implementation. The more the e-cash system meets the above requirements, the more secure the extended mobile agent clone detection system.

The transferable Ferguson's e-cash proposed by Wong [Won01] is one of the e-cash systems that meet the security requirements closely. It employs the RSA-based randomized blind signature, the polynomial secret sharing scheme and the secure one-way hash function. I and $\mathcal{K}$ of the agent can be adhered to the signature by a slight modification of the original signature scheme. It is a concrete example to demonstrate the feasibility to transplant an e-cash system to our mobile agent clone detection system, which will be explained in detail in the next chapter.

5.5. Attack Scenarios

To further elaborate the security aspect of our scheme, two possible attacks are introduced. The first one is the *chosen host response attack,* where the honest host could be traced from two co-related itinerary records. The second one is the *truncation and substitution attack*, where the cloning host may be wrongly identified by substitution of another host. These two attacks also apply to Chaum-Pedersen's general transferable e-coin.

5.5.1 The Chosen Host Response Attack

In our scheme, the migration protocol requires no reuse of the zero-value coin m for each migration of agent. It is important for preventing the chosen host response attack, as described by the following scenario:

1. There are two distinct agents $(I, \mathcal{K}, S)$ and $(I', \mathcal{K}', S')$. Both of them choose Host U_i as one of its visiting host.
2. Host U_i generates the challenge $c = f(m, \rho)$ with the same zero-value coin m and ρ,

upon receiving these two agents.

Host U_i constructs the response r with m while sending the agent $(I, \mathcal{K}, S)$ to Host U_{i+1}.

Host U_i constructs the response r' with the same m while sending the agent $(I', \mathcal{K}', S')$ to Host U_{i+1}'.

Since the challenges generated by Host U_{i+1} and Host U_{i+1}' are different, Host U_i gives two responses from different challenges corresponding to the same coin m, which is sufficient for extracting the information of U_i. As a result, the honest host is revealed, which violates itinerary privacy. (This attack also applies to Chaum-Pedersen's transferable e-coin, where untraceability is broken by spending two coins to the same user, who has multiple uses of the same zero-value coin on payment.)

Therefore, the challenge should be generated by an unused coin in the migration protocol to avoid such kind of attack.

5.5.2 The Truncation and Substitution Attack

For simplicity, our discussion so far is based on the assumption that an agent is not allowed to revisit the same host along its itinerary. However, if revisit of agent is considered, then the system is vulnerable to the truncation and substitution attack. Details of the attack are described as following.

1. An agent $(I, \mathcal{K}, S)$ carrying the coins $m_1, m_2, \ldots, m_\ell$, migrates to Host U. Host U sends away this agent with $m_1, m_2, \ldots, m_{\ell+1}$.
2. Then the agent travels around a number of hosts, and revisits Host U, with the coins $m_1, m_2, \ldots, m_\ell'$.
3. Host U truncates the itinerary records $m_{\ell+2}, m_{\ell+3}, \ldots, m_\ell'$ and sends away the agent with $m_1, m_2, \ldots, m_{\ell+1}$.

As a consequence, a portion of itinerary records is missing. If there is no cloning activity within this portion of itinerary, then no useful information is lost. Otherwise, the host we identified is just a substituted culprit of another host in the truncated loop, as illustrated by Figure 5.2.

In Figure 5.2, U_2 is the truncating host. At U_9, the original itinerary records received should be $m_1, m_2, \ldots, m_8$, follow by m_2' and m_9 (m_2 is different from m_2'). However, with truncation by U_2, the trimmed itinerary records become m_1, m_2, m_9.

If there is no cloning in the truncated loop, then the truncation does no harm since no

cloning host is escaped from prosecution.

If U_4 makes a clone, then from the clones at U_9 and U_5', we have (m_1, m_2, m_9) and (m_1, m_2, m_3, m_4, m_5'). Then we treat U_2 as the cloning host, while the actual cloning host is U_4.

If U_5 further makes the clone, then from the clones at U_5' and U_6', we have (m_1, m_2, m_3, m_4, m_5') and (m_1, m_2, m_3, m_4, m_5, m_6'). U_4 is identified as the cloning host, while U_5's cloning activity is not identified.

If U_7 further clones the agent, then from the clones at U_6' and U_8' we have (m_1, m_2, m_3, m_4, m_5, m_6') and (m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8'). U_5 is investigated, while not U_7.

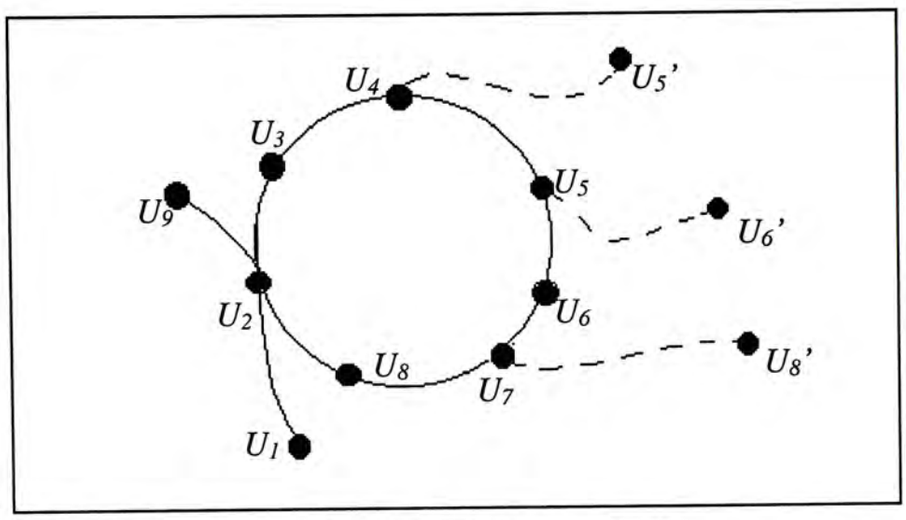


Figure 5.2: The truncation and substitution attack

From the above scenario, we can see that the last cloning host in the loop can always escape from investigation and its guilt is borne by the truncating host or another cloning host. (This attack also applies to Chaum-Pedersen's transferable e-coin, since the user can always recognize the coin when she sees it again. The last double spender in the loop can offend without being penalized.)

To deal with the truncation and substitution attack, a more sophisticated clone detection mechanism is required. However, it is not discussed in this paper.

5.6 An Alternative Scheme without Itinerary Privacy

Itinerary privacy is one of the most important benefits of our scheme. It is important for applications, such as the e-commerce, where the identity of a user is sensitive information. However, privacy is not always the concern in some other applications. We should choose the suitable mobile agent clone detection scheme according to the security requirement of particular

application's needs.

Baek's proposed an online scheme for detection of mobile agent clones [Bae98]. His scheme is an online scheme, which requires centralized oversight of every agent's action from a third party. As an online scheme, it takes the advantage that it can immediately stop the malicious action once the clone is detected. However, it produces extra communication overhead and limits privacy since the third party monitors every agent's behavior.

If privacy is a concern, then we prefer an offline scheme. However, there is a dilemma between offline migration and clone detection. We are required to keep secret the honest host identity while the identity of the cloning host must be disclosed. Our scheme makes use of the e-cash to provide such kind of conditional reveal. However, an offline scheme cause delayed halting of malicious action of agent clones.

Now we consider a lower security requirement: We are not concerned about itinerary privacy and we also tolerate delayed halting of malicious action, so now the scheme can be much more simplified.

In this section, we demonstrate a simple alternative of our offline clone detection scheme, based on consecutive executions of RSA-signatures [RSA78]. However, this alternative entirely gives up the itinerary privacy of the agent.

In this alternative scheme, we redefine an agent as

$$(I, \mathcal{K}, \mathcal{S}, \mathcal{T})$$

where I is the identifier of the agent, $\mathcal{K}$ is the code of the agent, $\mathcal{S}$ is the state of the agent, and $\mathcal{T}$ is the traveling history of the agent.

Furthermore, $\mathcal{T} = (\ell, \mathbf{U}, \mathbf{Y})$, with $\mathbf{U} = (U_1, U_2, \ldots U_\ell)$ and $\mathbf{Y} = (Y_1, Y_2, \ldots Y_\ell)$ where ℓ is an integer, $\mathbf{U}$ is an integer vector of the host identities with length = ℓ, $\mathbf{Y}$ is an integer vector of the signatures with length = ℓ.

Furthermore, the traveling history is valid if its parameters satisfy the following equations:

$$Y_0 = f(I, \mathcal{K})$$

and

$$Y_{i+1} = (Y_i)^{1/v_{i+1}} \quad \text{, for } 0 \le i \le \ell - 1$$

where v_i and $1/v_i$ are the RSA public key and RSA private key of host U_i respectively, and f is a public known one-way function.

In this alternative scheme, privacy is not a concern. Therefore, different from our scheme, host identities are transmitted in plaintext during migrations. Again, since privacy is not a concern, we do not need a bank or any exterior party to globally give a public key for verification of the traveling history. Instead, we use the public key of each visited host for this verification. Hence, no withdrawal or any preparation protocols are needed to initialize the signatures.

Following are the essential details of this alternative scheme.

Migration under the Alternative Scheme:

Step 1: The sending host U_{send} sends an agent (I, $\mathcal{K}$, $\mathcal{S}$, $\mathcal{T}$) to the receiving host U_{rec}.

Step 2: U_{rec} first checks that the traveling history $T = (\ell, \mathbf{U}, \mathbf{Y})$ is valid.

If yes, the receiving host computes a signature $Y_{rec} = (Y_\ell)^{1/v_{rec}}$, where $1/v_{rec}$ is the private key of U_{rec}.

(If no, then either U_{send} modified $\mathcal{T}$ or she embedded an invalid signature on $\mathcal{T}$. U_{rec} rejects the agent migration.)

Then she updates the old traveling history $\mathcal{T}$ to a new traveling history $\mathcal{T}' = (\ell' = \ell + 1, \mathbf{U}', \mathbf{Y}')$, where

$$\mathbf{U}' = \mathbf{U} \parallel U_{rec}$$

and $$\mathbf{Y}' = \mathbf{Y} \parallel Y_{rec}$$

Then it could execute the agent and get ready to send it to the next host.

Culprit Identification under the Alternative Scheme: Given two agents, with the same I and $\mathcal{K}$, we compare their traveling histories $\mathcal{T} = (\ell, U, Y)$ and $\mathcal{T}' = (\ell', U', Y')$. For $1 \le i \le \ell - 1$, if i is the first integer such that $U_i = U_i'$ but $U_{i+1} \neq U_{i+1}'$, then U_i is the cloning host.

For different mobile agent applications, we adapt different mobile agent clone detection schemes to fulfill their particular needs. If the immediate halting of malicious action is required, we can employ Baek's online scheme. Rather, if itinerary privacy is the main interest, our offline scheme based on general transferable e-cash can be used. If both of the above security requirements are not concerned, we can hire the alternative scheme based on consecutive executions of RSA signatures in this section, for the simplicity of implementation.

5.7 Conclusion Remarks

We introduced a detection scheme for unauthorized mobile agent cloning using Chaum-Pedersen's general transferable e-cash model. In the general transferable e-cash model, coins are transferred from user to user. Illegitimate duplicates and multiple uses of coins are detected and the identity of the user who performed the double spending is revealed from the duplicated coins. Similarly, a mobile agent migrates from host to host. One important objective of a mobile agent clone detection system is to identify the host who created unauthorized copies of the agent. Based on similar techniques, we can use any suitable e-cash system for the specific implementation of a mobile agent clone detection system.

No supervision from exterior party other than the sending host, the agent and the receiving host is required for each migration of agent, which reduces communication and protects the itinerary privacy. Clones will be detected after cloning occurs, and the cloning host identity will be revealed after investigation.

Two attacks to the scheme are introduced, both of which apply to Chaum-Pedersen's transferable e-cash. They are the chosen host responses attack and the truncation and substitution attack. Furthermore, we give a simple alternative to our scheme, which entirely gives up the itinerary privacy.

However, since the detection is made offline in our scheme, halting of the clone's malicious actions is delayed. Moreover, the growth in size of a transferred coin [ChaPe93] is an open issue for future research.

Chapter 6

Specific Implementation of the Mobile Agent Clone Detection System with Transferable Ferguson's E-coin

6.1 Introduction

In the previous chapter, we introduced a general mobile agent clone detection system based on the techniques of general transferable e-cash system. In this chapter, we give a specific implementation of the above general scheme by Wong's transferable extension [Won01] of Ferguson's single-term offline e-coin [Fer93]. Ferguson's scheme is an efficient e-cash system that meets the requirements of Chaum-Pederson's general e-cash model. Wong applies his scheme to Chaum-Pedersen's methods [ChaPe93] to make it transferable. We further transplant Wong's scheme to our general mobile agent clone detection system in this chapter.

Two agents with the same identity and the same fixed codes are unauthorized clones. The itineraries of two cloned agents can be used to reveal the identity of the host who performed the cloning. The two itineraries should have a common beginning, and then fork some time later. The host at the forking point is usually the culprit.

In our specific clone detection scheme under Ferguson's scheme, each agent records its itinerary in a file it carries called the *passport*. When two agent clones are captured, their passports reveal the identity of the culprit host.

In order to protect itinerary privacy, the passport is encrypted. The encryption is done such

that (1) whether or not an honest host is contained in an itinerary cannot be determined from the passport, and (2) the identity of culprit hosts can be computed from the clones' passports.

Each host is required to expend an *e-token* for the purpose of receiving, hosting, and sending away an agent. It conducts a multi-round interactive protocol to withdraw e-tokens from a supervisory body called *EPA* (*E-token and Passport Authority*). The use of e-tokens enforces that the hosts help agents record their itineraries honestly. The e-tokens can be withdrawn early and stockpiled. Then the migration of an agent from one host to another requires only offline computation involving the agent, the sending host, and receiving host but no other exterior entity.

The rest of this chapter is organized as follow: Section 6.2 defines the clone detection environment and gives basic assumptions. Section 6.3 shows the details of the protocols in our system. Security and privacy are analyzed in Section 6.4 while complexity analysis is given in Section 6.5. Finally, we have a chapter conclusion in the Section 6.6.

6.2 The Clone Detection Environment

In our mobile agent clone detection environment, there are hosts, mobile agents who migrate from hosts to hosts, a particular central host site called *EPA (E-token and Passport Authority)*. Each agent contains a file called *passport* as part of its state. When a host creates an agent, it needs to conduct an interactive protocol with the EPA to initialize the agent's passport. Each host obtains, through interactive protocols with the EPA, multiple files of the form *agent-hosting e-tokens*.

Assumptions about public parameters: The RSA public key v and the RSA modulus n of the EPA are known to all. The parameter v is assumed to be a large prime. The private key $1/v$ of the EPA is kept secret in the EPA. The parameter p is a large prime where p-1 is a multiple of n. Parameters g_1, g_2, g_3 are publicly known elements of large orders in the multiplicative group of Z_n^*. h_3 an h_4 are publicly known elements of orders n in the multiplicative group Z_p^*. The mappings f is a suitable one-way function from Z_n^* to Z_v^*, f_1 is a suitable one-way function from Z to Z_p^* with output of order n, f_2 is a suitable one-way function from $Z_n^* \times Z_n^*$ to Z_v^*, f_3 is a suitable one-way function from $Z_n^* \times Z_n^* \times Z_n^*$ to Z_v^*.

Each host has a unique identification number (identity), denoted U.

Throughout this paper, we assume computations are done modulo n unless otherwise specified.

Definition 1 An *agent* is defined as $\mathcal{A} = (I, \mathcal{K}, \mathcal{S}, \mathcal{P})$ where I is the identity of the agent, $\mathcal{K}$ is the code of the agent, $\mathcal{S}$ is the state of the agent, and $\mathcal{P}$ is the passport of the agent.

Remark: In classic notation, the agent consists only of I, $\mathcal{K}$ and $\mathcal{S}$, i.e. $\mathcal{A} = (I, \mathcal{K}, \mathcal{S})$. In that case, the passport $\mathcal{P}$ belongs to $\mathcal{S}$ and is a special part of $\mathcal{S}$.

Definition 2 A *passport* is a 7-tuple $\mathcal{P} = (\ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R})$ where ℓ is a positive integer, $\mathbf{a} = (a_1, a_2, \ldots, a_\ell)$, $\mathbf{b} = (b_1, b_2, \ldots, b_\ell)$, $\mathbf{c} = (c_1, c_2, \ldots, c_\ell)$ are integer vectors of length ℓ, and $\mathbf{x} = (x_1, x_2, \ldots, x_{\ell-1})$, $\mathbf{r} = (r_1, r_2, \ldots, r_{\ell-1})$, $\mathbf{R} = (R_1, R_2, \ldots, R_{\ell-1})$ are integer vectors of length $\ell - 1$. The *length of the passport* is ℓ.

The following technical definitions are useful.

Definition: An integer 6-tuple (a, b, c, x, r, R) is a *Type-I vector* if $R^v = C^r A^x B$, where $A = ag_1^{f(a)}, B = bg_2^{f(h_3^b)}, C = cg_3^{f(h_4^c)}$. An integer 8-tuple $(a, b, c, x, r, R, I, \mathcal{K})$ is a *Type-II vector* if $R^v = C^r A^x B$, where $h_1 = f_1(I)$, $h_2 = f_1(\mathcal{K})$, $A = ag_1^{f(a)}, B = bg_2^{f(h_1^b)}, C = cg_3^{f(h_2^c)}$

Definition 3 A passport $\mathcal{P}$ of length $\ell > 1$ is a *valid passport* for agent $\mathcal{A} = (I, \mathcal{K}, \mathcal{S}, \mathcal{P})$ if

(1) $(a_1, b_1, c_1, x_1, r_1, R_1, I, \mathcal{K})$ is of Type-II, and

(2) $(a_i, b_i, c_i, x_i, r_i, R_i)$ is of Type-I, for each i, $2 < i < \ell - 1$, and

(3) $x_i = f_3(a_{i+1}, b_{i+1}, c_{i+1})$, for each i, $1 < i < \ell - 1$.

By convention, all passports of length 1 are valid.

The passport structure of agent $(I, \mathcal{K}, \mathcal{S})$ is shown in Figure 6.1.

Definition 4 An *agent-creation e-token* for Host U and agent $\mathcal{A} = (I, \mathcal{K}, \mathcal{S}, \mathcal{P})$ is a 7-tuple of integers

$$(a, b, c, U, k, S, T)$$

satisfying $S = (C^k A)^{1/v}$, $T = (C^U B)^{1/v}$,

with $h_1 = f_1(I)$, $h_2 = f_1(\mathcal{K})$

$A = ag_1^{f(a)}, B = bg_2^{f(h_1^b)}, C = cg_3^{f(h_2^c)}$

Definition 5 An *agent-hosting e-token* for Host U is a 7-tuple of integers,

(a, b, c, U, k, S, T)

satisfying $S = (C^k A)^{1/v}$, $T = (C^U B)^{1/v}$

with $A = ag_1^{f(a)}, B = bg_2^{f(h_3^k)}, C = cg_3^{f(h_4^c)}$

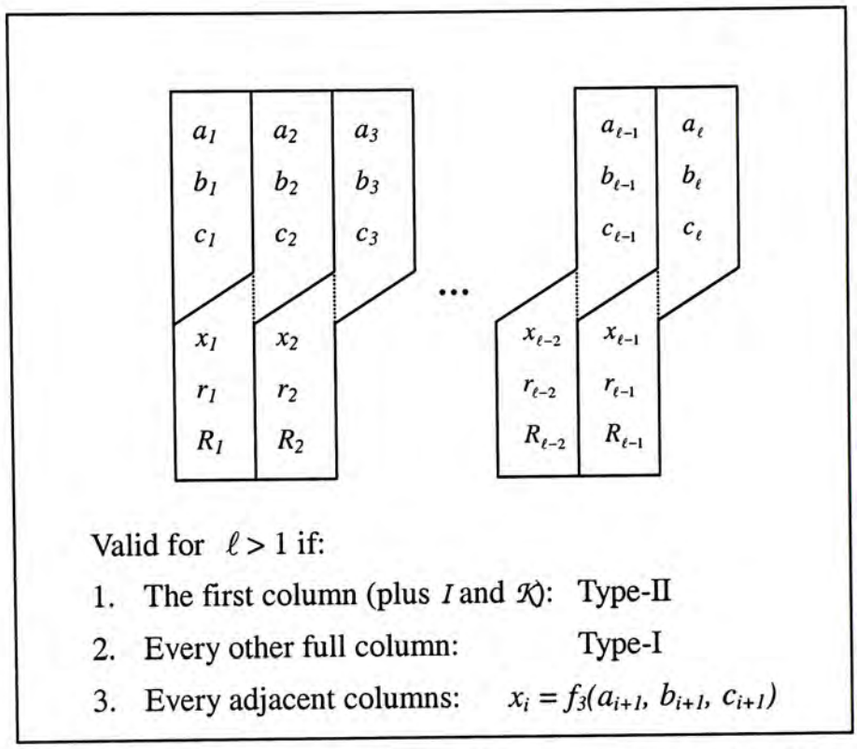


Figure 6.1: The passport structure of the agent (I, $\mathcal{K}$, S, $\mathcal{P}$) with length = ℓ

6.3 Protocols

6.3.1. Withdrawing E-tokens

A host interactively conducts these protocols with EPA for the withdrawal of e-tokens which will be needed for creating agents and hosting agents.

Agent-Hosting E-tokens Withdrawal Protocol

Host U obtains an agent-hosting e-token (a, b, c, U, k, S, T) from the EPA. This protocol is identical to the Withdrawal Protocol in Ferguson's e-coin [Fer93] (See Figure 4.3 for details). By setting Alice ← Host, Bank ← EPA, h_b ← h_3 and h_c ← h_4., we yield our protocol as shown in Figure 6.2.

The following summarizes the main features:

- Host presents its identity, U, to EPA.
- Host and EPA jointly generates the numbers a, b, c, k, S, and T, in such a way that these values are unknown to EPA.
- It is hard for Host to generate agent-hosting e-tokens without knowing $1/v$, the private key of EPA.

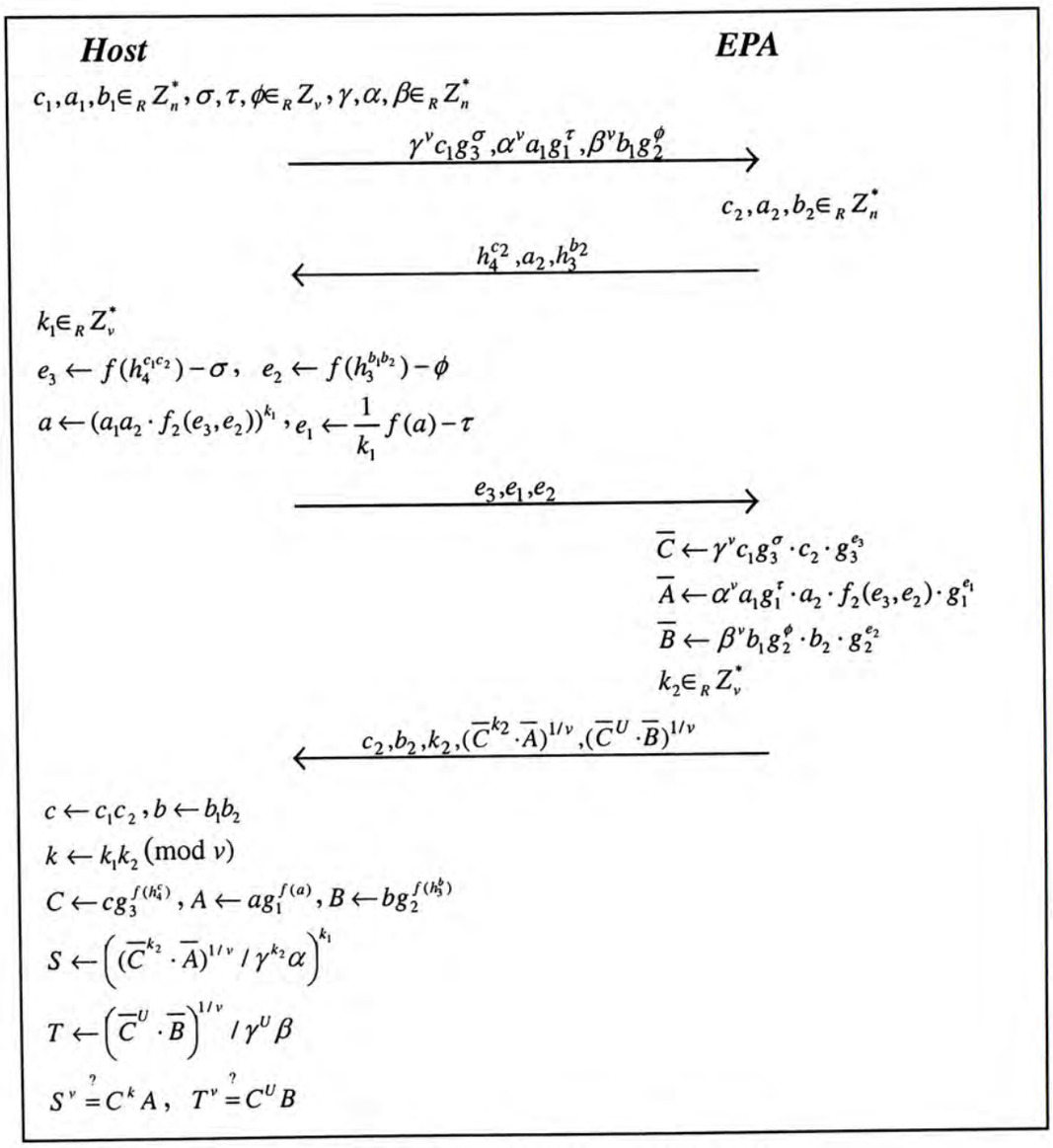


Figure 6.2: Agent-Hosting E-tokens Withdrawal Protocol

As we shall see below, a host needs to expend one agent-hosting e-token for each agent that travels through it.

A host repeats the protocol and obtains multiple (and distinct) agent-hosting e-tokens from the EPA.

Agent-Creation E-tokens Withdrawal Protocol

This protocol is almost identical to the Agent-Hosting E-tokens Withdrawal Protocol. Host U conducts an interactive multi-round protocol with EPA for the purpose of obtaining an agent-creation e-token for a new agent it just created, $\mathcal{A}$. Host needs I and $\mathcal{K}$ of the agent as the inputs to this protocol.

To modify Ferguson's withdrawal protocol (See Figure 4.3 for details) to the agent-creation e-token protocol, we set Alice ← Host, Bank ← EPA, $h_b \leftarrow h_1 = f_1(I)$ and $h_c \leftarrow h_2 = f_1(\mathcal{K})$. Ferguson requires h_b and h_c to be elements of order n in the multiplicative group Z_p^*. We can set $f_1(I) = h_3^{hash(I)} \pmod p$, where $hash(.)$ is a suitable one-way function mapping from I (and $\mathcal{K}$) to Z_n^*. Then h_1 and h_2 are qualified except negligible probability. The agent-creation e-token protocol is shown in Figure 6.3.

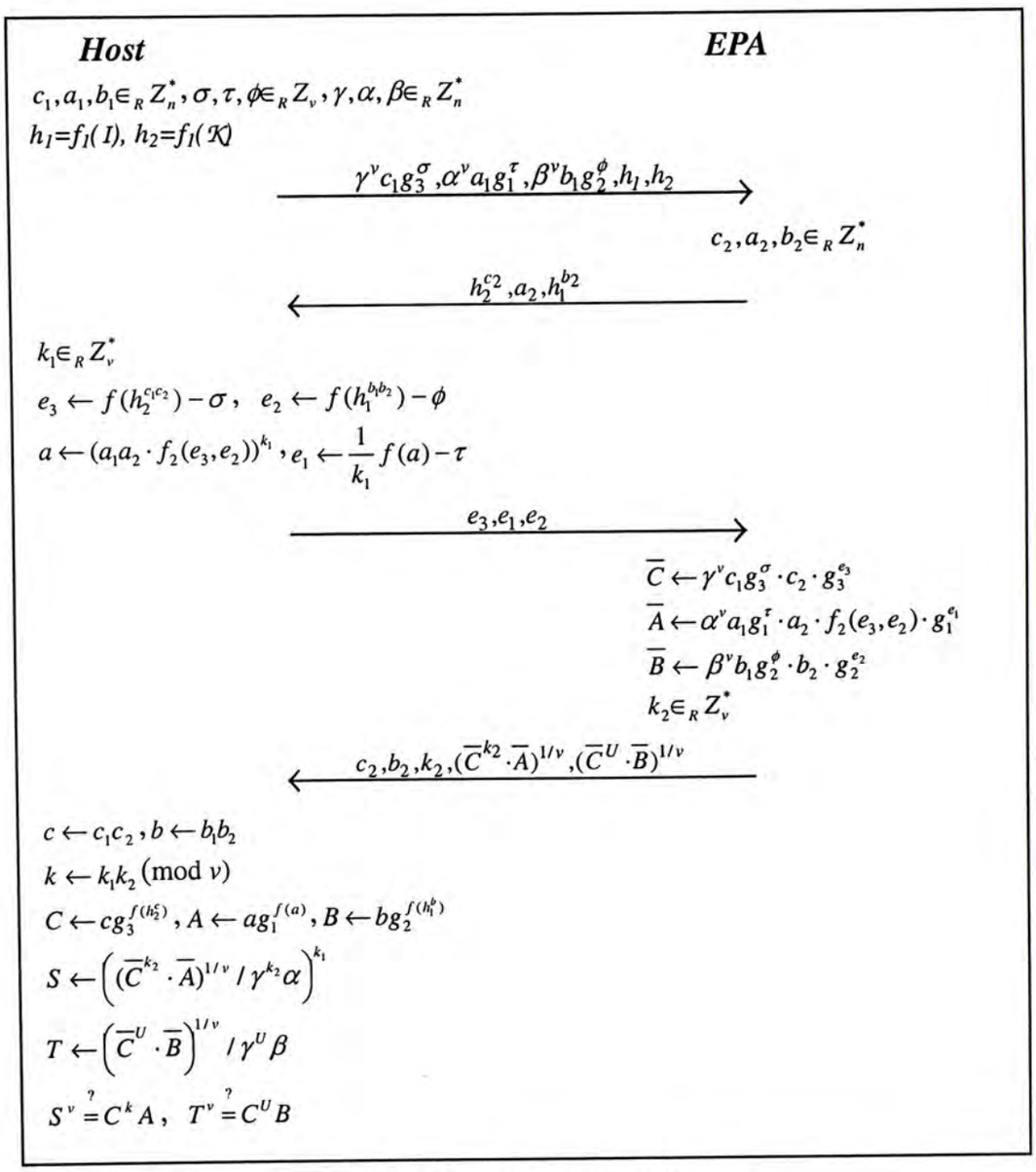


Figure 6.3: Agent-Creation E-tokens Withdrawal Protocol

EPA issues a single agent-creation e-token for each unique agent only, therefore, no agent is in procession of two different-agent creation e-tokens.

6.3.2 The Agent Creation Protocol

Host U creates a new agent via these steps:

Create: Host U creates the first three components of an agent $\mathcal{A} = (I, \mathcal{K}, S, \mathcal{P})$, where S is initialized.

Get Token: Then Host U performs Agent-Creation E-token Withdrawal with EPA, and obtains an agent-creation e-token (a, b, c, U, k, S, T) for $\mathcal{A}$.

Initialize Passport: Host U initializes $\mathcal{P} = (\ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R})$, with the agent-creation e-token, by setting $\ell = 1$, $a_1 = a$, $b_1 = b$, $c_1 = c$.

6.3.3. The Agent Migration Protocol

In this protocol, the agent $\mathcal{A} = (I, \mathcal{K}, S, \mathcal{P})$ migrates from the sending host U to the receiving host U', where $\mathcal{P} = (\ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R})$. After the protocol, the passport is updated to $\mathcal{P}' = (\ell' = \ell + 1, \mathbf{a'}, \mathbf{b'}, \mathbf{c'}, \mathbf{x'}, \mathbf{r'}, \mathbf{R'})$. Its length is incremented by one.

Host U' checks passport validity, perform challenge-and-response with U, and then "chops" the passport. The details of the steps are shown as below.

Verify Passport Validity: Host U' verifies that $\mathcal{P}$ is a valid passport for $\mathcal{A}$. Proceed if OK.

Challenge: Host U' selects an unused agent-hosting e-token $(a', b', c', U', k', S', T')$ in its possessions and issues a challenge x to Host U with $x = f_3(a', b', c')$.

Response: Host U computes, based on the e-token (a, b, c, U, k, S, T) which it used on $\mathcal{A}$ when $\mathcal{A}$ entered Host U, the following response (r, R) and send it to U': $r = xk + U \pmod{v}$, and $R = S^x T$.

[Exception: If Host U has just created the agent. i.e. $\ell = 1$, then the agent-creation e-token (a, b, c, U, k, S, T) is used instead of the agent-hosting e-token above.]

Verify Response: Host U' verifies that (a, b, c, x, r, R) is a Type-I vector. [Exception: If $\ell = 1$, then U' verifies that $(a, b, c, x, r, R, I, \mathcal{K})$ is a Type-II vector.]

Chop: Host U' "chops" (increments) the passport $\mathcal{P}$ to $\mathcal{P}'$ by setting $x_\ell = x$, $r_\ell = r$, $R_\ell = R$, $a_{\ell+1} = a'$, $b_{\ell+1} = b'$, $c_{\ell+1} = c'$.

Remark: Our scheme enforces that the same agent-hosing e-token is used for reception and send-away of the same agent. This is a crucial part of our security.

The updated passport from $\mathcal{P}$ to $\mathcal{P}'$ is shown in Figure 6.4.

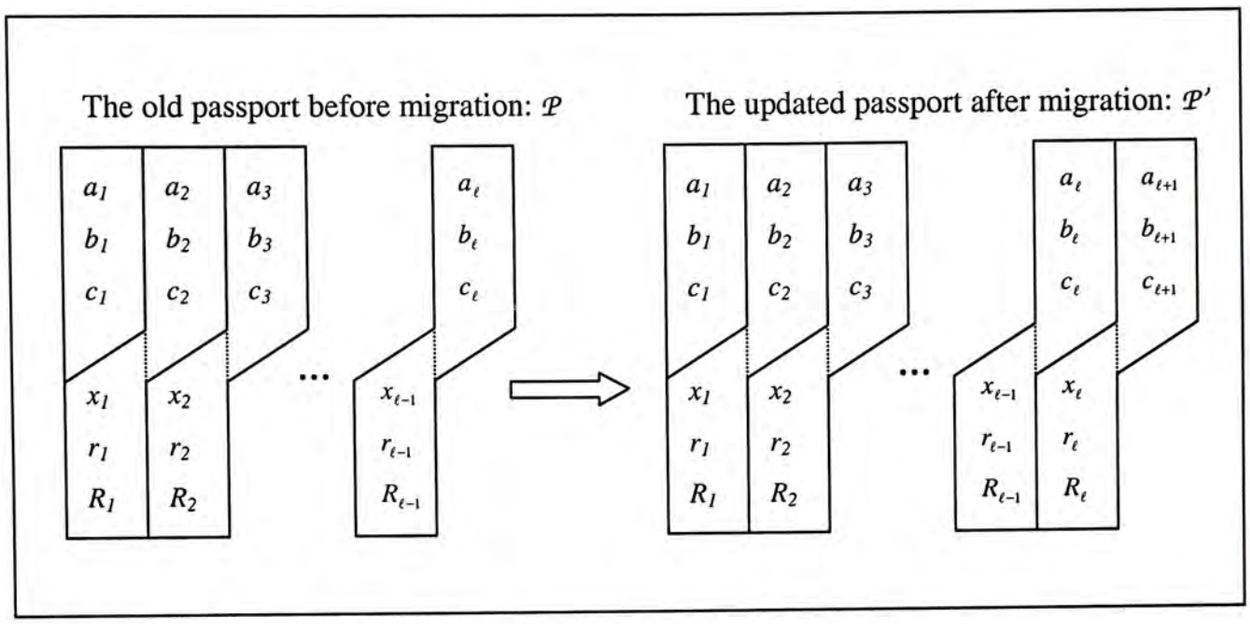


Figure 6.4: The passport update during $\ell'th$ migration of the agent

6.3.4 Clone Detection and Culprit Identification

Clone Detection Protocol. Two agents $\mathcal{A} = (I, \mathcal{K}, \mathcal{S}, \mathcal{P})$ and $\mathcal{A}'=(I', \mathcal{K}', \mathcal{S}', \mathcal{P}')$ are unauthorized clones if and only if $I = I'$ and $\mathcal{K} = \mathcal{K}'$. Any party can independently determine whether two given agents, whether just visiting its site or being transmitted from elsewhere for examination, are clones.

To identify the culprit, we need the following result:

Proposition 1: Setting aside a negligible probability, all e-tokens (including both agent-creation and agent-hosting) have unique headers (a, b, c).

Sketch of Proof: Ferguson's e-cash has unique headers. The agent-hosting e-tokens withdrawal is identical to Ferguson's e-cash withdrawal. The agent-hosting e-token withdrawal is a slight modification of it.

Culprit Identification Protocol

Given two clones $\mathcal{A}$ and $\mathcal{A}'$ with valid passports $\mathcal{P}$ and $\mathcal{P}'$, the host which performed the cloning can be computed as follows:

Remark: Clones without valid passports will not be able to travel onward. We consider them less harmful and do not investigate them in this paper. For simplicity, we also assume that no agent is allowed to revisit any host that it has traversed before.

Let i be the smallest positive integer such that $(a_i,b_i,c_i) \neq (a_i', b_i', c_i')$. We first focus on the case $i>1$. From Proposition 1, there is only one token (a, b, c, U, k, S, T) in our entire system

whose header $(a, b, c)= (a_{i-1}, b_{i-1}, c_{i-1})$. The host who withdrew this token from the EPA, U, is the culprit. His identity was privacy-protected but now it can be computed as follows. From the $(i-1)$-th entries of the passports $\mathcal{P}$ and we have $r_{i-1} = kx_{i-1} + U \pmod{v}$ and $r'_{i-1} = kx'_{i-1} + U \pmod{v}$. These two linear equations enable us to solve for U which is the identity of the culprit.

If i=1, then these two clones are still in the possession of their original agent-creating host and have never migrated yet.

If i does not exist and $\ell = \ell'$, then Host U who withdrew the token (a, b, c, U, k, S, T) with header $(a, b, c) = (a_l, b_l, c_l)$ made the clones but has not passed either of them out. If i does not exist and $\ell < \ell'$, then Host U who withdrew the token (a, b, c, U, k, S, T) with header $(a, b, c) = (a_\ell, b_\ell, c_\ell)$ made the clones but has not passed only one of them out. These cases are not considered as crimes.

Cloning Scenarios: To elaborate the culprit identification mechanism, we illustrate it by a scenario shown in Figure 6.5. In the scenario, multi-levels of cloning are investigated.

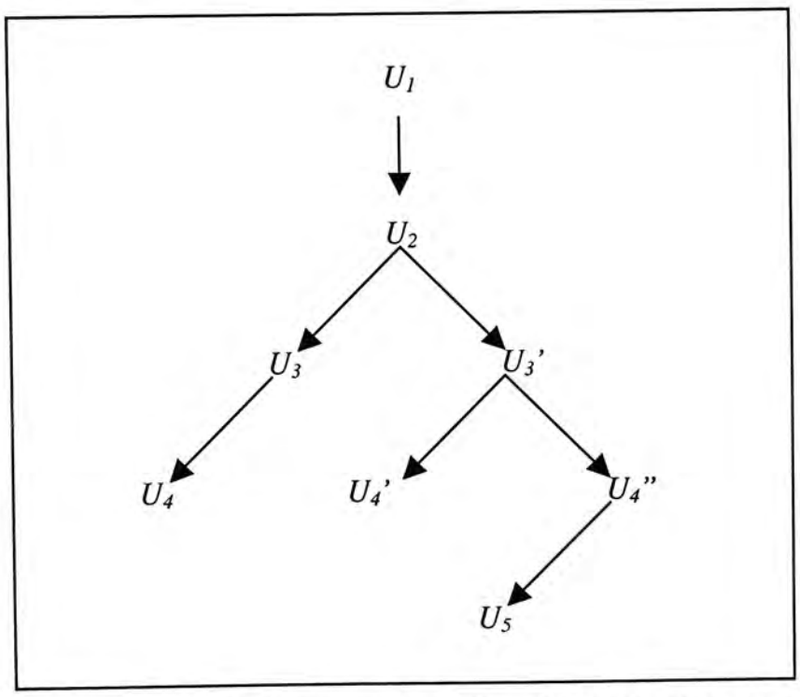


Figure 6.5: A cloning scenario with multi-levels of cloning

The host U_1 sends an agent to the host U_2. Then U_2 clones the agent and sends it to U_3 and U_3' respectively. U_3 then sends the agent to U_4, while U_3' further clones the agent and sends it to U_4' and U_4'' respectively. U_4'' further clones the agent and sends the clone to U_5. The paths of hosts visited by each clone are shown below:

Path 1: $U_1 \rightarrow U_2 \rightarrow U_3 \rightarrow U_4$
Path 2: $U_1 \rightarrow U_2 \rightarrow U_3' \rightarrow U_4'$
Path 3: $U_1 \rightarrow U_2 \rightarrow U_3' \rightarrow U_4'' \rightarrow U_5$

We express the passport at the end of each path as the following forms:

The passport $\mathcal{P}_1$ at the end of path 1:
$(a_1, b_1, c_1, x_1, r_1, R_1)(a_2, b_2, c_2, x_2, r_2, R_2)(a_3, b_3, c_3, x_3, r_3, R_3)(a_4, b_4, c_4)$

The passport $\mathcal{P}_2$ at the end of path 2:
$(a_1, b_1, c_1, x_1, r_1, R_1)(a_2, b_2, c_2, x_2', r_2', R_2')(a_3', b_3', c_3', x_3', r_3', R_3')(a_4', b_4', c_4')$

The passport $\mathcal{P}_3$ at the end of path 3:
$(a_1, b_1, c_1, x_1, r_1, R_1)(a_2, b_2, c_2, x_2', r_2', R_2')(a_3', b_3', c_3', x_3'', r_3'', R_3'')(a_4'', b_4'', c_4'', x_4, r_4, R_4)(a_5, b_5, c_5)$

Given the agent clones at host U_4 and U_4', we get $\mathcal{P}_1$ and $\mathcal{P}_2$. Since (a_3, b_3, c_3) and (a_3', b_3', c_3') is the first distinct pair, we know that U_2 is a cloning host by solving the equations:

$$r_2 = (k_2)(x_2) + U_2$$
$$r_2' = (k_2)(x_2') + U_2$$

Similarly, given the agent clones at host U_4' and U_5, we get $\mathcal{P}_2$ and $\mathcal{P}_3$. Since (a_4', b_4', $4_4'$) and (a_4'', b_4'', c_4'') is the first distinct pair, by interpolation between the two equations below, we realize that U_3' is the culprit who performed the clone.

$$r_3' = (k_3')(x_3') + U_3'$$
$$r_3'' = (k_3')(x_3'') + U_3'$$

From the above scenario, we can see that all malicious hosts could be eventually identified even if the agent is repeatedly cloned by different hosts within the itinerary.

6.4 Security and Privacy Analysis

We state several propositions concerning the security and privacy in our system.

Theorem 1: If $\mathcal{P}$ is a valid passport for $\mathcal{A}$, of any length, then $\mathcal{P}'$ is also valid passport for $\mathcal{A}$ provided the challenge-response is verified.

Proof. Straightforward.

Propositon 2. To produce an agent-hosting e-token without knowing $1/v$ is as hard as forging Ferguson's e-cash without $1/v$.

Proof Sketch: The two withdrawal protocols are identical.

Proposition 3. To produce a valid passport of length greater than one without knowing $1/v$ is as hard as forging Wong's transferable e-cash without $1/v$.

Proof Sketch: The passport is a modification of Wong's transferable e-cash. The only alterations are the passport initialization and the first agent migration. To cryptanalyze the alternations require inverting f_1, or cryptanalyze the agent-creation e-token.

Proposition 4. Assume Host U has not committed any unauthorized cloning. To determine whether U has been visited by a given agent is as hard as tracing whether Wong's transferable e-cash has passed through U', where U' is a party in Wong's system which has not committed e-cash forgery.

Proof Sketch: Similar arguments to Proposition 3.

Propositions 1 to 3, together with Theorem 1, assure the security of our system. Proposition 4 states that our system ensures itinerary privacy. The identities of traversed hosts cannot be determined from the passports unless the host errs.

You may refer to chapter 4 for more information about the security issues in the e-cash systems we reference to.

6.5 Complexity Analysis

6.5.1 Compact passport

As by Wong's implementation, we can modify our scheme to reduce the size of passport. The compact passport is redefined as

$$\mathcal{P} = (\ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{r}, R)$$

where the vectors (**x, r, R**) are reduced to vector **r** and a number R only.

Upon initialization of the passport, we set **r** = **0** and $R = 1$.

During migration, we do not store the challenge x_i in the passport, instead we compute it from the passport by

$$x_j = f_3(a_{j+1}, b_{j+1}, c_{j+1}) \qquad \text{for } 1 \le j \le \ell - 1$$

We validate the passport with $\ell > 1$ in a single step

$$R^v \stackrel{?}{=} \prod_{j=1}^{\ell-1} (C_j^{r_j} A_j^{x_j} B_j)$$

where $h_1 = f_1(I),\ h_2 = f_1(\mathcal{K})$

$$A_1 = a_1 g_1^{f(a_1)}, B_1 = b_1 g_2^{f(h_1^{b_1})}, C_1 = c_1 g_3^{f(h_2^{c_1})}$$

$$A_j = a_j g_1^{f(a_j)}, B_j = b_j g_2^{f(h_3^{b_j})}, C_j = c_j g_3^{f(h_4^{c_j})} \qquad \text{for } 2 \le j \le \ell$$

To update a passport from $\mathcal{P}$ = (ℓ, **a**, **b**, **c**, **r**, R) to $\mathcal{P}$ '= (ℓ ', **a'**, **b'**, **c'**, **r'**, R'), we set $R' = R \cdot R_{send}$, where R_{send} is the response from the sending host

The clone detection and the culprit identification remain the same.

6.5.2 Passport growth in size

For every migration, chops are added to the passport, which leads to the passport's growth in size. The accumulated traveling history will become an overhead for the agent transmissions and computations in the passport verification. The more hosts the agent visited, the larger the overhead. Intuitively, the growth in size of passport is not surprising, since the passport should contain some information of every visited host for future clone detection. And it is proved un-avoidable formally by [ChaPe93].

6.6. Conclusion Remarks

We introduced a mobile agent clone detection system based on the transferable e-cash. In our specific implementation, Wong's transferable extension of Ferguson's single-term offline

untraceable e-cash is used. The central idea is to mimic a paper passport, where host identity is "chopped" to an agent passport at each migration. Records on the passport are used for clone detection and culprit identification.

As an offline scheme, clones are detected after cloning has occurred. Therefore, halting of clone's malicious actions is delayed. Moreover, the passport grows in size as the agent travels [ChaPe93]. The complexity of the passport validity verification also grows.

Transferable offline untraceable e-cash schemes are notoriously complex. Our system inherits such high complexity. We have no current plan of implementation to test out system performance.

Chapter 7

Conclusions

In this thesis, we address the problems of unauthorized mobile agent cloning and propose a countermeasure by using Chaum-Pedersen's general transferable e-cash. Our scheme is generic in the sense that it can be implemented by a large class of the existing e-cash systems. In addition, a specific implementation of the general scheme by Wong's extension of Ferguson's single-term offline e-coin is presented. Our scheme benefits from offline computations and itinerary privacy.

To start with, we introduce the evolution of computing models, from the standalone computer era, to the mobile agent paradigm for the distributed computing. With its advantages in mobility and autonomy, it is regarded as a suitable model for applications such as the e-commerce and the information retrieval systems. Still, the mobile agent suffers from different kinds of security threats, causing its main deployment bottleneck. One of the security issues addressed by literatures is the unauthorized cloning of mobile agents.

Mobile agents can be duplicated with ease. If two duplicated mobile agents have the same identity, we call them unauthorized clones. Unauthorized cloning of mobile agent may cause multiple transactions that are unintended by the agent owner. On the other hand, the host can repudiate the transaction by falsely claiming the existence of clones. Baek therefore proposed an online detection scheme of agent clones based on the CP-Nets model. In his scheme, clones can be detected immediately, and the offender who performed the cloning can be identified. However, as an online scheme, centralized oversight from a third party is required. Continued monitoring of agent's activities imposes communication overheads and weakens the privacy of the agent.

We propose a new scheme for detecting mobile agent clones. Our scheme is motivated by the double-spending detection techniques in the transferable e-cash. An e-cash is an electronic counterpart of the physical cash in real life that serves as a medium of payment, a unit of

accounting and storage of value in electronic transactions. It is ultimately an electronic file that can be easily copied. The duplication and reuse of e-cash is called *double spending*. An e-cash must have a secure mechanism to deal with the problem of double spending. In many schemes, double spending cannot be prevented from happening. Rather, we detect it after its occurrence and identify the culprit so that it can be penalized. Similar to the mobile agent clone problem, illegitimate copies of the file is under investigation. We transplant the techniques to our mobile agent clone detection system.

We choose Chaum-Pedersen's general transferable e-cash model so that many classes of the existing e-cash systems can all apply to our mobile agent clone detection system. Important information of an agent is embedded into a coin when it is created. The agent carries the coin along the itinerary. For every host that the agent has visited, the host shares partial information of her identity to the agent in the form of a coin. A single instance of the coin hides the identity of the honest host, but multiple instances from cloned agents are sufficient to reveal the cloning host identity. In our scheme, migration only involves the agent, the sending host and the receiving host. Any other exterior supervision is not required. Moreover, the itinerary of the mobile agent is kept private. Only cloning hosts will be identified. However, as an offline scheme, there is a delay in halting the clone's malicious actions. Furthermore, the overhead of the coin grows after each migration.

In particular, we use Wong's transferable extension of Ferguson's single-term offline to implement the above general scheme. The idea is to mimic a paper passport, where the host identity is "chopped" to an agent passport at each migration. Records on the agent's passport are used for clone detection and culprit identification.

To conclude, although the mobile agent is not one of perfection, it is stepping forward. Some security issues regarded as difficult problems previously are now solved by various methods. This thesis proposes a solution to the problem of mobile agent cloning, which is addressed as an open issue in the prior literatures. We believe that, as more "security impossibilities" become possible, the mobile agent paradigm can realize its full potential and receive widespread deployment in the digital world.

Appendix

Papers derived from this thesis

[1] T. C. Lam and V. K. Wei, "A mobile agent clone detection system with itinerary privacy." Accepted by *IEEE 11th Int'l Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2002).*

[2] T. C. Lam and V. K. Wei, "Mobile agent clone detection using general transferable e-cash." Accepted by *2002 ACM International Symposium of Information Security (Infosecu02).*

Bibliography

[ANSI85] American National Standard Institute (ANSI), "ANSI X9.17-1985: Financial Institution Key Management." (1985).

[Bae98] J. Baek, "A design of a protocol for detecting a mobile agent clone and its correctness proof using Coloured Petri Nets." *Technical Report TR-DIC-CSL-1998-002, Info. & Comm., K-JIST,* (1998).
http://atom.kjist.ac.kr/~jsbaek/pub/tr-dic-1998-02.ps

[BGS98] S. Berkovits, J.D. Guttman, and V. Swarup, "Authentication for Mobile Agents." *In G. Vigna, editor, Mobile Agents and Security* (1998).

[BKR98] J. Bredin, D. Kotz, and D, Rus, "Market-based resource control for mobile agents." *Proc. of the Second Int'l Conf. on Autonomous Agents* (1998) 197-204.

[Bla79] G.R. Blakley, "Safeguard cryptographic keys." *Proc. of the AFIPS 1979 National Computer Conference* **48** (1979) 313 – 317.

[BMW98] I. Biehl, B. Meyer and S. Wetzel, "Ensuring the Integrity of Agent-Based Computations by Short Proffs." *In Kurt Rothermel, Fritz Hohl (Eds), Mobile Agents* (1998) 183-194.

[Bra93] S. Brands, "Untraceable Off-line Cash in Wallet with Observers." in *Advances in Cryptology-CRYPTO' 93* (1993) 302-318.

[Cha82] D. Chaum, "Blind Signatures for Untraceable Payments." *In Advances of Cryptology: Proceedings of CRYPTO' 82* (1982) 199-203.

[Cha85] D. Chaum, "Security without Identification: Transaction Systems to make big Brother Obsolete." *In Communications of the ACM* **28:10** (1985) 1030-1044.

[Cha92] D. Chaum. "Randomized blind signature." *Personal communications* (1992).

[ChaPe93] D. Chaum and T. P. Pedersen, "Transferable Cash Grows in Size." I*n Advances in Cryptology –EUROCRYPT' 93* (1993) 390 – 407.

[Che98] D. M. Chess, "Security Issues in Mobile Code Systems." In *Mobile Agents and Security. LNCS 1419* (1998) 1-14.

[CEvdG88]D. Chaum, J.H. Evertse, and J. van de Graff, "Demonstrating possession of a discrete logarithm without reveling it." *In Advances in Cryptology – EUROCRYPT'87* (1988) 127-141.

[CFN88] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash." In *Advances in Cryptology – CRYPTO'* 88 (1988) 319 – 327.

[CLS94] W. Caelli, D. Longley, and M. Shain, *Information Security Handbook.* Macmillan Publisher (1994).

[DH76] W. Diffie and M.E. Hellman, "New directions in cryptography." IEEE *Transactions on Information Theory* **22** (1976) 644-654.

[Fan01] X. Fan, "On Splitting and Cloning Agents." *Turku Centre for Computer Science, TUCS Technical Reports, No 407* (2001).

[Fer93] N. Ferguson, "Single Term Off-line Coins." In *Advances in Cryptology - EUROCRYPT' 93* (1993) 318-328.

[FG96] S. Franklin and A. Graesser, "Is It an Agent, or just a Program? A Taxonomy for Autonomous Agents." *In Proceedings of the Third International Workshop on Agent Theories, Architectures, and Language.* (1996).
http://www.msci.memphis.edu/franklin/AgentProg.html

[FGS96] W. M. Farmer, J. D. Guttman, and V. Swarup, "Security for mobile agents: Issues and requirements." *Proc. of the 19th National Information Systems Security Conf.* (1996) 591 – 597.

[Fon99] P. W. L. Fong, "Proof Linking: Modular Verification of Mobile Programs in the Presence of

Lazy, Dynamic Linking."
ftp://fas.sfu.ca/pub/cs/techreports/1999/

[FPV98] A. Fuggetta, G. P. Picco and G. Vigna, "Understanding Code Mobility." *In IEEE Transactions on Software Engineering* **24** (1998).

[GM00] M. J. Grimley and B. D. Monroe, "Protecting the Integrity of Agents: An Exploration into Letting Agents Loose in an Unpredictable World." *ACM Crossroads*, July 2000.
http://www.acm.org/crossroads/xrds5-4/integrity.html

[HCK95] C.G. Harrison, D.M.Chess and A. Kershenbaumn, "Mobile Agents: Are they a good idea?" *IBM Research Repor,* (1995)
http://www.research.ibm.com/iagents/paps/mobile_idea.pdf

[Hoh98] F. Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts." *In G. Vigna, editor, Mobile Agents and Security* (1998).

[Jan99] W. A. Jansen, "Countermeasures for Mobile Agent Security." (1999)
http://www.itl.nist.gov/div893/staff/jansen/ppcounterMeas.pdf

[Jen92a] K. Jensen, "Coloured Petri Nets: Basic Concepts, Analysis Method and Practical Use Volume1." *EATCS Monographs on Theoretical Computer Science* (1992).

[Jen92b] K. Jensen, "Coloured Petri Nets: Basic Concepts, Analysis Method and Practical Use Volume2." *EATCS Monographs on Theoretical Computer Science* (1992).

[JK99] W. Jansen and T. Karygainnis, "NIST Special Publication 800-19 -- Mobile Agent Security." (1999).
http://csrc.nist.gov/mobileagents/publication/sp800-19.pdf

[KAG98] G. Karjoth, N. Asokan and C. Gulcu, "Protecting the Computation Results of Free-Roaming Agents." *Second International Workshop on Mobile Agents* (1998).

[KLO97] G. Karjoth, D.B. Lange and M. Oshima, "A Security Model for Aglets." *IEEE Internet Computing* **1:4** (1997) 68 – 77.

[Kob87] N. Koblitz, "Elliptic Curve Cryptosystems." *Mathematics of Computation* **48:177** (1987) 203-209.

[KR98] R. Kumanduri, C. Romero, *Number Theory with Computer Applications*. Prentice-Hall, Inc. (1998)

[Lai92] X. Lai, "On Design and Security of Block Ciphers." *ETH Series Information Processing* **(1)** (1992).

[LO99] D.B. Lange, M. Oshima, "Seven good reasons for mobile agent. Dispatch your agents; Shut off your machine." *Communications of the ACM* **42:3** (1999) 88-89.

[MOV97] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, Fla (1997).

[Mil85] V. S. Miller, "Use of Elliptic Curve in Cryptography." *Advacnes in Cryptology – Proceedings of CRYPTO'85* (1985).

[NBS77] National Bureau of Standard, "Federal Information Processing Standard (FIPS), Publication 46: The Data Encryption Standard." (1977)

[NC99] S. K. Ng and K. W. Cheung, "Protecting Mobile Agents against Malicious Hosts by Intention Spreading." *Proc. Int. Conf. On Parallel and Distributed Processing Techniques and Applications (PDPTA' 99)* (1999) 725-729.

[NIST88] National Institute of Standards and Technology (NIST), "Federal Information Processing Standard (FISP) Publication 46-1: Data Encryption Standard." (1988)

[NL98] G. C. Necula and P. Lee, "Safe, Untrusted Agents Using Proof-Carrying Code." *In G. Vigna, editor, Mobile Agents and Security* (1998).

[OO91] T. Okamoto and K. Ohta, "Universal electronic cash." In *Advances in Cryptology – CRYPTO' 91* (1991) 324 – 337.

[vA90] H. van Antwerpen, *Off-line electronic cash*. Master's Thesis, Eindhoven University of Technology, Department of Mathematics and Computer Science (1990).

[RSA78] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signature and public-key cryptosystems." *Communications of the ACM* **21** (1978) 120-126.

[SBH96] M. Straser, J. Baumann, and F. Hohl, "Mole – a Java based mobile agent system." *In 2nd ECOOP Workshop on Mobile Object Systems*, (1996) 28 – 35.

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

[Sch98] B. Schneier, "The Secret Story of Nonsecret Encryption." (1998) http://www.ddj.com/documents/s=909/ddj9875b/9875b.htm

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

[She99] O. Shehory, "Spawning information agents on the web." *Intelligent Information Agents, M. Klusch (Ed.)* (1999).

[Sim92] G. J. Simmons, "Contemporary Cryptology: The Science of Information Integrity." IEEE Press (1992).

[SS97] M. Strasser and M. Schwehm, "A Performance Model for Mobile Agent Systems." *In H. Arabnia (ed.), Proc. Int Conf. On Parallel and Distributed Processing Techniques and Applications (PDPTA'97), CSREA* **2** (1997) 1132-1140.

[SSCJ98] O. Shehory, K. Sycara, P. Chalasani, and S. Jha, "Agent Cloning: an approach to agent mobility and resource allocation." *IEEE Communications* **36:7** (1998) 58-67.

[ST98] T. Sander and C. F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts." *In G. Vigna, editor, Mobile Agents and Security* (1998).

[Sti95] D. R. Stinson, *Cryptography: theory and practice*. CRC Press, Inc. (1995).

[Sun98] T. Sundsted, "An Introduction to agents." (1998) http://www.javaworld.com/javaworld/jw-07-1998/jw-06-howto.html

[Won01] H. Y. Wong, *Issues in Electronic Payment Systems: A New Off-line Transferable E-coin Scheme and a New Off-line E-check Scheme*. Master's Thesis, Dep. of Information Engineering, The Chinese University of Hong Kong (2001).