University of Wollongong
Research Online

University of Wollongong Thesis Collection

University of Wollongong Thesis Collections

2011

Policy-based secure agents

Yikai Wang
University of Wollongong

Recommended Citation

Wang, Yikai, Policy-based secure agents, Master of Computer Science thesis, University of Wollongong. School of Computer Science and Software Engineering, University of Wollongong, 2011. http://ro.uow.edu.au/theses/3261

Research Online is the open access institutional repository for the University of Wollongong. For further information contact Manager Repository Services: morgan@uow.edu.au.

NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Policy-based Secure Agents

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

Master of Computer Science

from

UNIVERSITY OF WOLLONGONG

by

Yikai Wang

School of Computer Science and Software Engineering
May 2011

© Copyright 2011

by

Yikai Wang

All Rights Reserved

Dedicated to
My mother and my father

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Yikai Wang
May 16, 2011

Abstract

Software agents, as the main body of agent-oriented E-commerce systems, have a great potential for E-commerce applications. Agents can dynamically discover and compose E-services and mediate interactions, which can serve as delegates in order to handle routine tasks, monitor activities, set up contracts, execute business processes, and find the best services [Gri01, GK94, Nwa96]. However, agents are vulnerable to several attacks from the running environment, such as, insecure network, malicious users and malicious hosts, etc. Therefore, the success or failure of this paradigm is connected to the issues whether the mechanisms in agents can protect their security and privacy. Policy-based cryptography, which consists of two cryptographic primitives: policy-based encryption and policy-based signature, was firstly proposed by Bagga and Molva [BM06, BM05]. It has the advantage of addressing the security problems of software agents. We are motivated by the concept of policy-based cryptography [BM06, BM05] and use cryptographic primitives mentioned above to provide security and privacy for agents systems.

There are several kinds of software agents, such as mobile agents, multi-agents, intelligent agents and distributed agents. In this thesis, we focus on mobile agents and multi-agents. Mobile agents have the ability of migrating across different execution environments. Due to this property, the security and privacy of mobile agents can be easily compromised when they are traveling on a hostile environment. Therefore, security and privacy are critical to mobile agents applied in E-commerce. There exist several solutions for mobile agent security [ST98b, KBC00, LKK01, ZMZ09], among which the proxy-based model is believed to be a sound solution to provide authentication, where the visiting host of an agent acts as a proxy signer who can sign an offer. However, any host (including a malicious host) could act as the signer to forge valid signatures. To solve this problem, a secure policy-based mobile agent scheme is introduced in this thesis, that is, only the hosts who satisfy the designated policies can generate a valid signature. A security model and a rigorous

security proof related to scheme are provided as well.

Multi-agent systems are different from mobile agents systems in the way that they do not have the property of mobility. In most of current multi-agent systems, the identity authentication and privacy issues have not drawn adequate attentions. Without the protection mechanism, the secure information exchange channel can not be provided in multi-agent systems. Policy-based schemes can solve the problem of multi-agents identity authentication and privacy protection. Bagga and Molva [BM06] firstly proposed a concrete policy-based encryption (PBE) scheme. PBE allows entitiesd to encrypt a message with respect to a credential-based policy so that only the entities who are compliant with policy can successfully decrypt the ciphertext. However, the size of ciphertext in [BM06] will increase linearly if the policy set is growing larger. This is a significant issue in a resource limited communication channel. In this thesis, we propose a new policy-based encryption scheme with much smaller ciphertext size and less computational cost. To the best of our knowledge, this is the first time PBE is applied to the multi-agent system, which allows policy-based authenticated communications and provides privacy protection to agents. We define a rigorous security model for multi-agent transaction, which captures the most powerful attacks including adaptive chosen message attacks. The security of our scheme is based on the hardness of ((p, g, F)-GDDHE) problem in the random oracle model.

Acknowledgement

I thank all who have helped to make this thesis possible. First of all, I would like to express my heartfelt gratitude to my supervisors, Associate Professor Yi Mu and Associate Professor Minjie Zhang, who provide me the chance to do research on agents security. I also appreciate their patient guidance, invaluable instructions, detailed explanations and insightful criticism.

I am also indebted to my beloved parents. Their concern and comfort always cheer me up when I encounter any problem.

I would like to express my appreciation to Mr. Fuchun Guo and Dr. Wei Wu, for their time, patience and critical comments on my research. I am grateful to my friendly schoolmates and housemates as well. They help me a lot no matter what in life, or in study and provide me with helpful discussions, suggestions and information.

My thanks also go to those scholars whose books and articles I have made reference in my thesis. The administrative staff and technical support staff of School of Computer Science and Software Engineer, Faculty of Informatics should also be appreciated for their supports.

Yikai Wang

Wollongong, March 2011

Publications

Yikai Wang, Yi Mu and Minjie Zhang. Policy-based authentication for mobile agents. In Proceedings of the 7th Information Security Practice and Experience Conference (ISPEC2011). Guangzhou, China, 2011.

Yikai Wang, Yi Mu and Minjie Zhang. Policy-based encryption with much smaller ciphertext and application in multi-agent systems. (draft)

Contents

Abstract **v**

Acknowledgement **vii**

Publications **viii**

1 Introduction **1**

1.1 Mobile Agent and Security Issues 1
1.2 Multi-Agent and Security Issues 3
1.3 Related Work 5
1.4 Challenging Issues and Motivations 7
1.5 Proposed Solutions 9
1.6 Overview of the Thesis 9

2 Background **11**

2.1 Preliminaries 11
2.1.1 Bilinear Map 11
2.1.2 Random Oracle Model 11
2.2 Number-Theoretic Problems 12
2.2.1 The Discrete Logarithm Problem 12
2.2.2 The Diffie-Hellman Problem 13
2.2.3 The Bilinear Diffie-Hellman Problem 13
2.2.4 The General Diffie-Hellman Exponent Assumption 14
2.3 Hash Functions 14
2.4 Public Key Encryption 15
2.5 Identity-based Encryption 16
2.6 Broadcast Encryption 17

2.7 Digital Signatures . . . 18
2.8 Proxy Signature . . . 19
2.9 Ring Signature . . . 20
2.10 Secret Handshake . . . 21
2.11 Hidden Credentials . . . 21
2.12 Oblivious Signature-based Envelopes . . . 22
2.13 Policy-based Cryptography . . . 23
2.14 Attribute-based Cryptography . . . 24

3 Policy-Based Authentication for Mobile Agents 26
3.1 Introduction . . . 26
3.2 Mobile Agent Model . . . 28
3.2.1 Policy Model . . . 28
3.2.2 Mobile Agent Model . . . 29
3.3 Definition and Security Model . . . 30
3.3.1 Notation . . . 30
3.3.2 Definition . . . 31
3.3.3 Security Models . . . 32
3.4 Policy-based Authentication for Mobile Agent Scheme . . . 35
3.5 Security Analysis . . . 37
3.5.1 Existential Unforgeability Against $\mathcal{A}_1$ . . . 37
3.5.2 Existential Unforgeability Against $\mathcal{A}_2$ . . . 40
3.5.3 Existential Unforgeability Against $\mathcal{A}_3$ . . . 43
3.5.4 Credentials Ambiguity . . . 45
3.6 Summary . . . 46

4 Policy-based Encryption with Much Smaller Ciphertext and Application in Multi-agent Systems 47
4.1 Introduction . . . 47
4.2 Definition . . . 50
4.2.1 Policy, Assertion and Credential . . . 50
4.2.2 A High Level Description of Our Scheme . . . 50
4.2.3 Definition of PBE Scheme . . . 51
4.2.4 Security Model . . . 52
4.2.5 Pairing and Complexity Assumption . . . 53

4.3 Our Scheme 54
4.4 Efficiency 55
4.5 Security Analysis 56
4.6 Summary 58

5 Conclusion **60**

Bibliography **62**

List of Tables

4.1 Bandwidth comparison between two schemes 56
4.2 Computational costs in terms of pairing computation comparison between two schemes 56

List of Figures

1.1 Malicious Host Threat Classification [BC02] 4

3.1 Mobile agent model 30

4.1 Multi-agent model 51

Chapter 1

Introduction

The introduction and development of the software agent technology is the combination of artificial intelligence and network techniques. The complexity of computer software architecture and organizational structure is continually growing in Internet which is the largest network environment at present. Meanwhile, the trend of software development is distribution and intelligential because the traditional software design approach dissatisfies the practical needs any more. The purpose of distributed software is to disperse the tasks, and to seek the resolutions by the knowledge-shared software modules or network nodes. In the meantime, the purpose of intelligent is to coordinate the intelligent behaviors among intelligent hosts. Both of the purposes become the basic idea of software agent. The agent is an entity, which is operated in dynamic environment and enjoys the close control ability. It acts on behalf of other entities and exhibits some level of the key attributes of learning, co-operation and mobility [Hoh8b].

1.1 Mobile Agent and Security Issues

Mobile agent is a software entity which can transfer from one host to another one in the heterogeneous network environment automatically, and can interact with other agents or resources. Mobile agent is a special software agent, which not only enjoys the basic features including autonomy, reactivity, productivity and reasoning of agents, but also possesses mobility. The mobility of mobile agents refers to traveling between several hosts and finishing appointed tasks on behalf of users. Due to the mobility, the new computer model reduces network load dramatically and improves the communication efficiency.

Based on outstanding properties stated above, mobile agent systems are useful for electronic commerce systems. Mobile agents can be used to search offers from

different stores in the internet, negotiate the transaction autonomously with the sellers, pay for the goods and collect a receipt as proofs. Obviously, mobile agents play a significant role in e-commerce [MSS99]. Therefore, the success or failure of this paradigm is directly connected to the issues whether the security mechanisms in mobile agent systems are sufficient to prevent malicious behaviors from the untrusted networks. Chess [Che98] presented the reason why mobile agent systems will raise new security issues. Since mobile code and data can be accessible and completely controlled by a host during its execution, it is impossible for a mobile agent to carry any sensitive secret without disclosing it to a malicious host.

In the existing literature, there are different security aspects [Che98, Jan99, Jan00, LM99] of mobile agents presented as follows:

Insecure Networks. As mentioned before, mobile agents travel in open network environment, which is insecure in the real world. The common security issues in the network will also interfere the activities of entities. Those activities include migrating from one host to another, the communicating with other agents and users. The best way to protect them is to provide a secure communication channel to reach the goal of data integrity and data confidentiality in cryptography.

Malicious Agents. In a mobile agent system, there exist some agents, who could hold malicious intentions to attack other agents or hosts. For example, in a mobile agent-based flight ticket system, a user will release a mobile agent to search a host who represents a flight ticket agency. Malicious agents can steal the target agent's task, tamper with the offer information, or obstruct from host execution. It is clear that other agents and hosts should be protected from such attacks. Volpano and Smith [VS98] proposed a concept that allows agents run in a sandbox environment, which limits the agents' privileges. The agent will be granted extra privileges if it is trusted by the host.

Malicious Users. Not all users are legal in open mobile agent system. A malicious user can also launch attacks against the system, like eavesdropping the communication channel, creating and releasing malicious agents to attack other agents and hosts. In the real e-commerce system, malicious users could create malicious agents to steal and tamper with the secret which carried by targeted agents, impersonate the targeted agents to communicate with hosts. They could also deny releasing an agent and refuse the payment made by the agent. Obviously, it is also desirable to protect agents and hosts from these malicious users.

Malicious Hosts. It is possible and easy to protect hosts from malicious agents

by using sandbox mechanisms [VS98], however, protecting agents against malicious hosts seems to be a difficult and even impossible task. During the process that a mobile agent is executed on a host, the relationship between the agent and host is unfair. The agent can not prevent the secret key from disclosing it to the host. To demonstrate the problem of malicious hosts, Hohl [Hoh8b] presented a model in which the following attacks by malicious hosts were identified:

- Code and execution integrity: Can a mobile agent protect itself against tampering by a malicious host?
- Code privacy: Can a mobile agent conceal the program it wants to execute?
- Computing with secrets in public: Can a mobile agent remotely sign a document without disclosing the private key?

Bierman and Cloete [BC02] summarize four malicious host attack categories, illustrated in Figure 1.1: integrity attacks, availability refusal, confidentiality attacks, and authentication risks. Integrity and confidentiality alterations reveal and exploit the private information contained in the code and dynamic agent state. Together with authentication risks, these attacks represent attempts by a malicious party to gain unfair advantage without explicitly refusing agent execution. These host types represent the worst-case mobile agent could risk.

Fairness of contract. The basic requirement of fair contract is non-repudiation of both parties. In other word, the host should not be able to deny an offer that has been promised and the customer should not be able to deny a service he wished to obtain [ZMZ09]. For example, after the booking process of the flight ticket is finished with customer's payment, the host can repudiate this offer he promised and refuse to deliver the flight ticket [LKK01]. Therefore, providing the fairness of contract in a mobile agent-based e-commerce system for both hosts and users is also important.

1.2 Multi-Agent and Security Issues

A multi-agent system (MAS) consists of several artificial agents, which are also code programs, interact or work together to perform certain tasks over a standard communication network. Several multi-agent systems are available as commercial products and also have been implemented in various research projects.

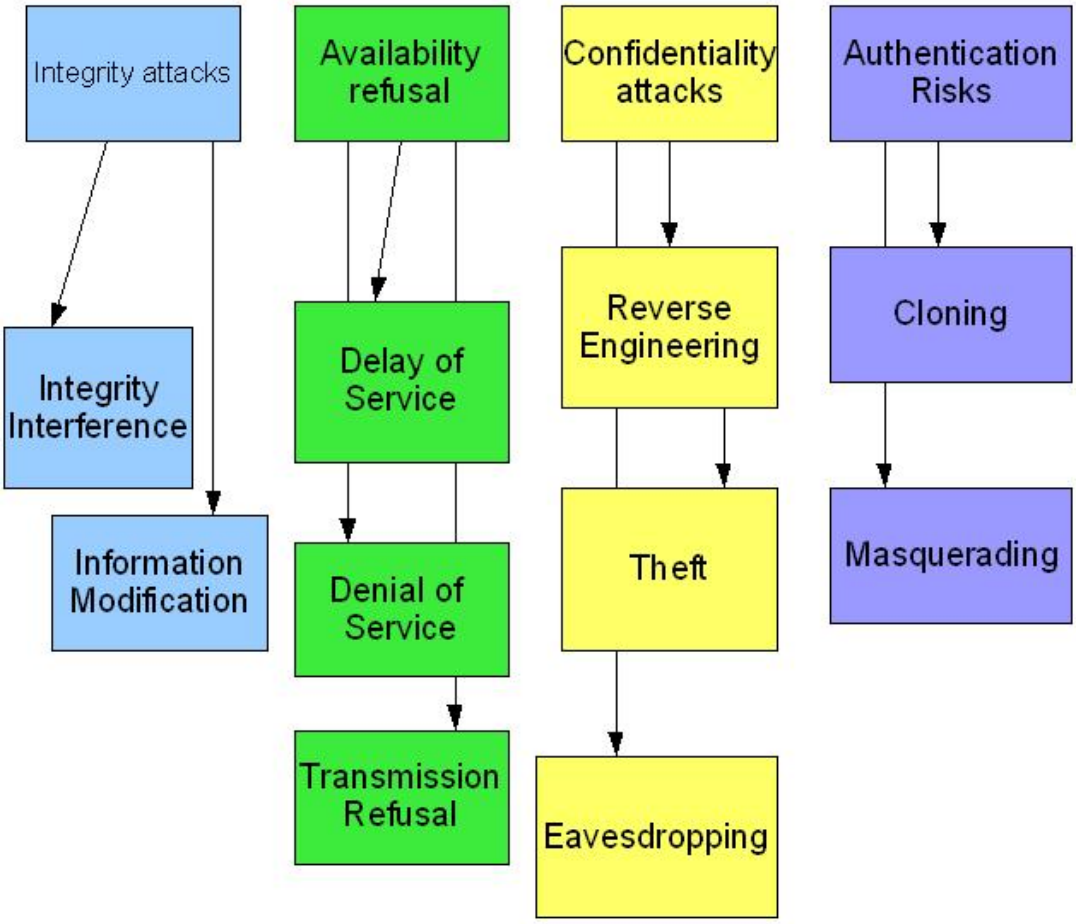

Figure 1.1: Malicious Host Threat Classification [BC02]

Multi-agent system is considered as an e-business solution in open service environment. In a multi-agent system, on behalf of their owners, software agents interact with each other to complete a task together. In order to fulfill the common task coherently, the agents must communicate among themselves. Therefore, multi-agent systems face security issues when it is deployed in open environment. Poslad *at el.* [PCC03] pointed out that considering the secure communicating issues for multi-agent systems, the secure goals of confidentiality, data integrity, authentication of origin and availability should be reached.

1.3 Related Work

There are a number of solutions proposed to solve the problem of code and execution integrity [Yee99, LM99]. Basic cryptographic primitives have been used to implement their solutions, such as asymmetric encryption, digital signature. Taking data that is given to a mobile agent and the identity of next host as input strings into a hash function; the host signs the output of the hash function; the generated signature is then encrypted by the mobile agent's public key. However, these protocols have some flaws which have been pointed out by Roth [Rot01], hosts' authority are disclosing in these protocols and any attacker can use protocols to generate a large number of data, which leads to the consequence that the protocol security objectives are completely compromised. Meanwhile, it is difficult for a mobile agent owner to detect the tampering behaviors that an attacker has forged and replaced a part of the data.

Vigna [Vig97, Vig98] proposed a mechanism to protect the mobile agents from untrusted environment, which allows an agent owner to detect any possible attempt of tampering with agent data, code and state by using cryptographic traces that are logs of the operations performed by an agent during its lifetime after agent termination. However, complete execution traces are costly to transmit since the agent owner has to perform the execution again by himself.

Kotzanikolaou *et al.* [KCK99] proposed a secure mobile agent system on untrusted hosts environment if they do not include payment capability. In their system, there exist a master agent and multiple mobile agents. The master agent does not travel, while mobile agents can travel, but only to negotiate with one particular host, and they can not complete a transaction. Mobile agents return the master agent with contracts signed by the hosts. The master agent evaluates the signed contracts and presents the results to the user. However, even if mobile agents do not have signing ability, they are still potentially vulnerable from attacking by malicious hosts. Because it just solves the problem of computing with secrets in public, while the problem of code and execution integrity remains the same. So the solution in [KCK99] are certainly still valuable [CPV03].

The security issues in agent execution are not only focus on privacy, but also have to be performed in suitable environment. Riordan and Schneier [RS98] proposed a concept called environmental key generation in which cryptographic keys are constructed from certain environmental data. The key is used to encrypt an

agent or part of it so that it would only be decrypted and executed if this environmental data were present at the host. Their scheme could prevent agents from being executed on a malicious host, provide host identity authentication according to the environmental conditions in theory [CPV03].

In order to solve the problem of code privacy and computing with secrets in public, Sander and Tschudin [ST98a] proposed a concept called encrypted functions protocol to provide execution privacy. This protocol guarantees the encrypted functions and the original functions are both executable. The general model in mobile agents system is as follows: The user (the agent owner), Alice, has a function f, the remote host, Bob, has input data x; Alice wants to compute $f(x)$ by releasing a mobile agent to Bob; however, Alice does not want Bob to learn anything about f and Bob does not want to reveal x to Alice either. The protocol allows that Alice encrypts f with a key k, and sends $E(f)$ to Bob. Bob calculates $(E(f))(x)$, and returns the result to Alice. At last, Alice calculates $f(x)$ using k with which she encrypted f [CPV03].

Considering the real mobile agents system for secure electronic transactions, the remote host needs to sign an offer which is embedded in the mobile agent. It raises a problem: how to execute the sign process without revealing the signature function to host and without revealing the host private key that is used to sign the offer to others? With respect to this issue, Sander and Tschudin [ST98a] then proposed undetachable signatures based on encryption functions. Considering the signature routine s and the function f that produces the output to be signed, these two functions should be combined together, which is $f_{signed} = s \cdot f$. The mobile agent has f and f_{signed}. The host can calculate $f(x)$ and f_{signed}. The host should only be able to construct $s(n)$ with $n = f(x)$, but unable to construct $s(m)$ for an arbitrary m. However, the secure implementation was not provided in [ST98a].

Although undetachable signatures [ST98a] can successfully solve the problem of protecting mobile agents from malicious hosts, there is no concrete scheme proposed in [ST98a]. Kotzanikolaou *et al.* [KBC00] firstly implemented the notion of undetachable signatures by using RSA. Let $h = hash(C, req_C)$, where C is the identity of the user (owner of the mobile agent), and req_C denotes user's requirements and constraints. Let $f(.) = h^{(.)} \mod n$, be the function that produces the output which has to be signed. Combined with the signature function, $k = h^d \mod n$ (d is the private key of the user), we have $f_{signed(.)} = k^{(.)} \mod n$. Let $x = hash(S, C, bid_S)$, with S the identity of the server (the host), and bid_S the server's bid, the transaction will be

$m = f(x)$, and $s(m) = f_{signed}(x)$. In other words, the mobile agent actually already carries a message/signature pair (r_C, $s(r_C)$), from which, it is easy to calculate another message/signature pair, containing both the host's bid and user constraints $(r_C^{xs}, s(r_C^{xs}))$. Unfortunately, the host cannot generate arbitrary message/signature pairs [CPV03].

The undetachable signature conceals the customer's private key, which can securely sign a contract with remote hosts, but it does not provide host non-repudiation since the undetachable signature does not contain host's signature. Proxy signatures can be adopted here to represent both customer and host signatures, it can also provide the fairness of the contract. Lee, Kim and Kim [LKK03] proposed a scheme to solve the host non-repudiation problem. Mobile agent is one of the applications of proxy signatures, the original signer (a user) has to delegate the signing capability to mobile agents or remote hosts. The proxy signature is used to allow all hosts to generate valid signatures after executing the mobile agent [KBC00, SL03, PL01a, LKPY06, LKK01, KJLK01]. Although it ensures the fairness of the contract, all hosts, including malicious ones in the network could use the method in [LKK03] to generate valid signatures. It raises another security issue in mobile agent systems, that is, delegation abuse.

Recently, Zhang, Mu and Zhang [ZMZ09] proposed a secure mobile agent scheme to achieve host authentication with designated hosts, where only selected hosts can be included in the agent network. In their scheme, they added the ID information in the designated host list that allows certain hosts to perform an agent task. Therefore, the host non-repudiation problem and delegation abuse problems can be solved. The host should not be able to deny an offer that has been promised by himself and the customer should not be able to deny a service he wishes to obtain; and only the selected host can sign the contract later.

In terms of multi-agent system, unfortunately, most existing research only pay attention to the security protection of the message exchanged among agents. There is few work focusing on authentication and privacy issues of multi-agents systems in the literature.

1.4 Challenging Issues and Motivations

Although the scheme in [ZMZ09] solves all the problems related to security issues of mobile agents, it is still insufficient to satisfy all needs in real world. The scheme

can be regarded as a simple policy-based scheme, where there is only one policy, that is, the ID information in their authentication system. We consider a scenario, when a user wants to buy a flight ticket, he may consider the following factors about the ticket provider: (1) Who can apply the targeted ticket? It is usually disguised by hosts' identities. The host who does not belong to flight ticket agency should be excluded from the network; (2) The location of the flight ticket agency; (3) The reputation of the flight ticket agency. However, we found that there is no such protocol in mobile agents which can solve this problem. This raises a challenging question: How to authorize designated hosts that satisfy a set of policies and can generate a valid policy-based signature in hostile environment? We are motivated by cryptographic primitive called policy-based signature scheme, which can address the problem of host authentication with designated host [BM05]. A policy specifies the constrains under which a specific action can be performed on a certain sensitive resource [BM06, BM05, TSM09b, TSM09a].

Few work has been done to successfully resolve multi-agent system security and privacy protection issues, which inspired us to provide a security solution to this problem. Based on some existing cryptographic primitives: Secret Handshakes [BDS$^+$03, CJT04], Hidden Credentials [HBSO03, BHS04], Oblivious Signature-based Envelop (OSBE) [LDB03, NT05] and Policy-based Encryption (PBE) [BM05], the problems in multi-agent systems seem to be solved by any one of these four cryptographic primitives. However, the two entities in secret handshake must be authenticated to each other by using both credentials [BDS$^+$03]. Holt, Bradshaw and Seamons [HBSO03] proposed hidden credentials to solve the problem brought from secret handshake, which allows the sender to encrypt the message only using the receiver's credentials. But they did not give a concrete scheme. OSBE enables a sender to encrypt a message and sends the result to a server who can decrypt the message if and only if the receiver has the proper credentials. However, it is not suitable for our system due to the following reasons: OSBE requires senders and receivers to have agreement of the credentials that receivers will use to decrypt sender's ciphertext. It can not be avoided that senders encrypt a message to receivers without disclosing credentials he must use to decrypt the ciphertext.

1.5 Proposed Solutions

In order to response to the challenging issues mentioned in Section 1.4, we propose two secure schemes as follows:

First, we provide a sound answer to the question of delegation abuse in mobile agent systems by introducing a novel scheme, which allows designated hosts to perform an agent task. To the best of our acknowledge, it is the first proxy policy-based signature scheme. The list of designated hosts can be chosen by the agent owner according to the set of policies. In the proposed scheme, hosts are treated as proxy signers who hold a set of credentials satisfying the policies embedded in mobile agents so that they are allowed to generate a valid "proxy" signature. Our scheme not only solves the privacy issue in mobile agent systems, but also the issue of host non-repudiation.

In order to solve security issues in multi-agent systems, we propose a new novel policy-based encryption scheme with much smaller ciphertext size and apply it into the multi-agent system. In our scheme, we require that a sender encrypts a message with a selected set of policies. Any receiver, as long as he/she has the corresponding credentials, can decrypt the ciphertext. Hence, the scheme provides both secure policy-based authenticated communications and privacy protection to agents. We use the Key Encapsulation Mechanism (KEM) in our scheme so that long messages can be encrypted with a short session key. Our credential system is based on the short signature scheme described in [BLS04]; our policy-based encryption scheme extends the ID-based broadcast encryption defined in [Del07]. We manipulate the policies that are formalized as monotonic logic expressions involving complex disfunctions of conditions which is more realistic for agents. Compared with the current policy-based encryption scheme proposed by Bagga and Molva [BM06], our scheme is more efficient. In our solution, the size of ciphertext is much smaller, and the private key is linear in the size of policy.

1.6 Overview of the Thesis

This thesis concentrates on the security issues in mobile agent systems and multi-agent systems, proposes the solutions to each systems. The rest of this thesis is organized as follows:

Chapter 2 presents the background knowledge in cryptography we have to use

to this thesis, such as bilinear map, random oracle and hard number-theoretic problems; some cryptographic primitives include hash functions, public key encryption, identity-based encryption, *et al*.

In Chapter 3, we investigate an authentication problem that relates to mobile agents interacting with hosts within open and hostile environment. Particularly, what we want is to allow only an authorized host to execute a mobile agent and subsequently generate a signature, as a proof of the host's environment. We attempt to achieve this through a cryptographic scheme that allows only designated hosts, which are compliant to a set of policies, to generate valid signatures. Associated security model and proofs are also presented.

A new policy-based encryption scheme for multi-agent systems is proposed in Chapter 4, which achieves secure communication within agents. Compared with current policy-based encryption schemes, the size of ciphertext is much smaller and the computational cost is lower in our scheme. Associated security model and proofs are also presented.

At last, we conclude the thesis in Chapter 5.

Chapter 2

Background

2.1 Preliminaries

2.1.1 Bilinear Map

Bilinear Map: In mathematics, a bilinear map is a abstract map which is combining a pair of group elements to an element of another group [MvOV97].

Definition 2.1 *Let G_1 be a cyclic additive or multiplicative group, and G_2 be a cyclic multiplicative group. Let p be a generator of G_1. A function $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map if it satisfies the following properties:*

- Bilinearity: additive group: $e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q)$ and $e(P, Q_1 + Q_2) = e(P, Q_1) \cdot e(P, Q_2)$ or $e(Pa, Qb) = e(P, Q)ab$; or multiplicative group: $e(P_1 \cdot P_2, Q) = e(P_1, Q) \cdot e(P_2, Q)$ and $e(P, Q_1 \cdot Q_2) = e(P, Q_1) \cdot e(P, Q_2)$ or $e(P^a, Q^b) = e(P, Q)^{ab}$;
- Non-degenerate: There exists $P \in G_1$ and $Q \in G_1$ such that $e(P, Q) \neq 1$;
- Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

A bilinear map satisfies above three properties above is said to be an admissible bilinear map. Weil or modified Tate pairings on an elliptic curve are used to implement bilinear maps [Jou02].

2.1.2 Random Oracle Model

Hash functions are widely used in many cryptosystems. An ideal hash function needs to possess special properties such as one-wayness and strong randomness. In

practice, random oracles are introduced to typically used for modeling cryptographic hash functions in schemes where strong randomness assumptions are needed [BR93]. A random oracle outputs a response corresponding to each query it receives. For the same queries, the random oracle will response with the same output. Random oracle is a mathematical abstraction used in cryptographic proofs.

2.2 Number-Theoretic Problems

In modern cryptography, the intractable mathematical problems are the basis of the security proof for most cryptosystems. We say a mathematical problem is intractable if no one has found an efficient (in polynomial time) algorithm to solve it. Public-key cryptography is based on the intractability of certain mathematical problems. Early public-key systems, such as the RSA algorithm, are secure assuming that it is difficult to factor a large integer composed of two or more large prime factors. For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly-known base point is unfeasible. The size of the elliptic curve determines the difficulty of the problem. Hereafter, we only describe the intractable problems which will be used in this thesis. For more information about the intractable problems the readers may refer to [MvOV97].

2.2.1 The Discrete Logarithm Problem

In abstract algebra, the discrete logarithm [McC90, CEdG88] is a group-theoretic operation that is analogous to the ordinary logarithm of real numbers or complex numbers. For example, an ordinary logarithm $\log_a(b)$ is a solution of the equation $ax = b$ over the real or complex numbers. Similarly, if g and h are elements of a finite cyclic group G then a solution x of the equation $g^x = h$ is called a discrete logarithm to the base g of h in the group G. The discrete logarithm problem is considered as an intractable problem in mathematics and cryptography [COS86, CS03] because there is no efficient (in polynomial time) algorithm that can solve it.

Definition 2.2 *Given a cyclic group G and its generator g, for any element $a \in G$, the discrete logarithm problem is to find an integer x where x $(0 \leq x \leq |G|)$ satisfying $a = g^x$.*

2.2.2 The Diffie-Hellman Problem

The Diffie-Hellman problem is considered as one important intractable problem in cryptography, which was firstly proposed by Whitfield Diffie and Martin Hellman [DH76]. The Diffie-Hellman problem is closely related to the discrete logarithm problem. It is generally believed that the Diffie-Hellman problem can be solved in polynomial time if solving the discrete logarithm problem is easy in polynomial time.

Many variants of the Diffie-Hellman problem have been proposed. The most notable variant is called the Decisional Diffie-Hellman (DDH) problem [JN]. The Diffie-Hellman problem is usually referred to the Computational Diffie-Hellman (CDH) problem in order to distinguish with the DDH problem. It is obvious that if there exists an algorithm which can solve the CDH problem in polynomial time, then the DDH problem can also be solved in polynomial time.

Definition 2.3 *Given a cyclic group G over a finite field, a generator g of G, and two elements g^x, g^y of G where $x, y \in Z_p$, p is a prime number, the Computational Diffie-Hellman (CDH) problem is to compute g^{xy}.*

Definition 2.4 *Given a cyclic group G over a finite field, a generator g of G, and two elements g^x, g^y, A of G where $x, y \in Z_p$, p is a prime number, the Decisional Diffie-Hellman (DDH) problem is to decide whether A equals to g^{xy}.*

2.2.3 The Bilinear Diffie-Hellman Problem

The Bilinear Diffie-Hellman (BDH) Problem is as one variant of the Diffie-Hellman problem over the bilinear map, which consists of the Decisional Bilinear Diffie-Hellman (DBDH) problem and the Gap Bilinear Diffie-Hellman (GBDH) problem. The security of some cryptographic schemes is based on the BDH such as in [BF01, Jou00] since these problems are widely believed to be computationally hard. The DBDH was introduced in [BB04], and the GBDH was introduced in [CC03, OP01].

Definition 2.5 *Given cyclic groups G_1, G_2 over a finite field, a generator g of G_1, a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, and three elements $g^x, g^y, g^z of G_1$ where x, y, z are three unknown integers, the Bilinear Diffie-Hellman (BDH) problem is to find the element $e(g, g)^{xyz}$ of G_2.*

Definition 2.6 *Given cyclic groups G_1, G_2 over a finite field, a generator g of G_1, a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, and three elements $g^x, g^y, g^z of G_1$ where x, y, z are three unknown integers, and an element $A \in G_2$, the Decisional Bilinear Diffie-Hellman (CBDH) problem is to decide whether A equals to $e(g, g)^{xyz}$.*

Definition 2.7 *Given cyclic groups G_1, G_2 over a finite field, a generator g of G_1, a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, and three elements $g^x, g^y, g^z of G_1$ where x, y, z are three unknown integers, the Gap Bilinear Diffie-Hellman (GBDH) problem is to find the element $e(g, g)^{xyz} of G_2$ with the help of the $DBDH$ oracle.*

2.2.4 The General Diffie-Hellman Exponent Assumption

The generalization of the Diffie-Hellman exponent assumption was proposed by Boneh, Boyen and Goh [BBG05]. They introduced a class of assumptions including DDH, BDH, q-BDHI and q-BDHE assumptions.

Definition 2.8 *((f, g, F)-GDDHE). let $\mathcal{B} = (p, G_1, G_2, G_t, e(., .))$ be a bilinear map group system and let f and g be two coprime polynomials with pairwise distinct roots, of respective orders t and n. Let g_0 be a generator of G_1 and h_0 a generator of G_2. Solving the (f, g, F)-GDDHE problem consists, given*

$$g_0, g_0^{\gamma}, ..., g_0^{\gamma^{t-1}}, \qquad g_0^{\gamma \cdot f(\gamma)}, \qquad g_0^{k \cdot \gamma \cdot f(\gamma)}$$

$$h_0, h_0^{\gamma}, ..., h_0^{\gamma^{2n}}, \qquad h_0^{k \cdot g(\gamma)},$$

and $T \in G_T$, in deciding whether T is equal to $e(g_0, h_0)^{k \cdot f(\gamma)}$ or to some random element of G_T [Del07].

2.3 Hash Functions

A hash function is a mathematical function which compress an input of arbitrary length to a result with a fixed length. In cryptography, hash functions are generally used for gene rating message digests or message fingerprints to achieve data integrity, realizing a ciphertext correctness verification mechanism, and used as practical pseudo-random functions. Hash function is a deterministic function which converts arbitrary length messages to fixed length strings.

Definition 2.9 *Let H denote a hash function whose fixed output length n, that is*

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

can be regarded as a hash function. The hash functions must satisfy the following properties:

- *one wayness: Given a value y, it is hard to find a value x such that $H(x) = y$.*
- *Weak collision resistance: Given a value x, it is hard to find a value $x^{'} \neq x$ such that $H(x) = H(x^{'})$.*
- *Strong collision resistance: It is hard to find a pair $(x, x^{'})$ where $x \neq x^{'}$ such that $H(x) = H(x^{'})$.*

2.4 Public Key Encryption

In cryptography, the encryption scheme includes private key (symmetric key) and public key (asymmetric key) encryption schemes. In a private key (symmetric key) encryption scheme, both the sender and receiver use the same key to perform encryption and decryption process. If this key is disclosed, communications will be compromised. In a public key encryption scheme, there exists a public/private key pair (PK, SK). The sender uses receiver's public key PK to encrypt the message and the receiver user his private key SK to decrypt the ciphertext.

Definition 2.10 *A public key encryption scheme consists of three algorithms: KeyGen, Enc and Dec.*

KeyGen: Take the system secure parameter 1^k as input, the key pair generation algorithm outputs a pair of public key and private key.

$$(pk, sk) \leftarrow KeyGen(1^k)$$

Enc: Take a message m and the public key pk, the encryption algorithm outputs a ciphertext c.

$$c \leftarrow Enc(m, pk)$$

Dec: Take the ciphertext c and the private key sk as input, the decryption algorithm outputs the encrypted message m if and only if the input satisfy $c \leftarrow Enc(m, pk)$ and $(pk, sk) \leftarrow KeyGen(1^k)$.

An asymmetric-key cryptosystem was published in 1976 by Whitfield Diffie and Martin Hellman [DH76]. Asymmetric key algorithms were publicly announced to be developed by James H. Ellis, Clifford Cocks, and Malcolm Williamson at the Government Communications Headquarters (GCHQ) in the UK, they also developed Diffie-Hellman key exchange and a special case of RSA. Since the 1970s, a large number and variety of encryption, digital signature, key agreement, and other techniques have been presented in the field of public-key cryptography.

2.5 Identity-based Encryption

Identity-based encryption (IBE) belongs to a family of public key encryption, which eliminates the need of a public key infrastructure (PKI). A Sender can use receiver's unique identity, such as receiver's e-mail address, as the public key to encrypt the message. If the receiver wants to decrypt the ciphertext, he must authenticate himself to a Private Key Generation (PKG) Center from where he can get the corresponding private key related to the identity (public key). PKG uses its master secret key to generate the private key for a user.

Definition 2.11 *An identity-based encryption (IBE) scheme consists of three entities: PKG, sender and receiver. And it is specified by four algorithms: Setup, Extract, Encrypt, Decrypt [BF01]:*

- *Setup: Take a security parameter k as input, this algorithm returns system parameters $params$ and master key. The system parameter is publicly accessible, while the master key is only known to the Private Key Generator (PKG).*
- *Extract: Take $params$, master key and arbitrary identity ID as input, the algorithm returns a private key d. Here, ID is an arbitrary string that is used as a public key, and d is the corresponding private decryption key. The Extract algorithm extracts a private key from the given public key.*
- *Encrypt: Take $params$, ID, and a message M as input, the algorithm returns a ciphertext C.*
- *Decrypt: Take $params$, ID, C and a private key d as input, the algorithm returns a M.*

The concept of IBE was first proposed by Shamir [Sha84] in 1984, but the first IBE scheme was constructed by Boneh and Franklin until 2001 [BF01], which is based on elliptic curves with bilinear pairings under Bilinear Diffie-Hellman assumption.

2.6 Broadcast Encryption

Broadcast Encryption, firstly introduced by Fiat and Naor in [FN94], is the foundation based on which a broadcaster can encrypt and transmit data to users who are listening to a broadcast channel, meanwhile they use the private key to perform decryption. During the process of encryption, the set of identities chosen by the broadcaster will help the receiver to decrypt ciphertext.

Definition 2.12 *A broadcast encryption system includes four randomized algorithms [GW09]:*

- *Setup(n,l): Take a security parameter k, the number of receivers n and the maximal size $l \leq n$ of a broadcast recipient group as input, the algorithm outputs the system parameters $params$ and a public/secret key pair (PK, SK).*

- *KeyGen(i,SK): Take an indx $i \in \{1, ..., n\}$ and the secret key SK as input, the algorithm outputs a private key d_i.*

- *Enc(S,PK): Takes a subset $S \subseteq \{1, ..., n\}$ and a public key PK as input, If $|S| \leq l$, the algorithm outputs a pair (Hdr, K) where Hdr is called the header and $K \in \mathcal{K}$ is a message encryption key.*

  Let ε_{sym} be a symmetric encryption scheme with key-space $\mathcal{K}$, and algorithms $SymEnc$ and $SymDec$. Let M be a message to be broadcast to the set S, and let $C_M \leftarrow SymEnc(K, M)$ be the encryption of M under the symmetric key K. The broadcast to users in S consists of (S, Hdr, C_M).

- *Dec(S,i,d_i,Hdr,PK): Take a subset $S \subseteq \{1, ..., n\}$, an index $i \in \{1, ..., n\}$, a private key d_i for i, a header Hdr, and the public key PK. If $|S| \leq l$ and $i \in S$ as input, then the algorithm outputs the message encryption key $k \in \mathcal{K}$. The key K can then be used to decrypt C_M to obtain M.*

Later, Naor *et. al* [NNL01] presented a fully collusion secure broadcast encryption system that is efficient for broadcasting to all but a small set of revoked users.

Their scheme can be used to encrypt to $n - r$ users with a header size of $O(r)$ elements and private key of size $O(log^2 n)$. After that Halevy and Shamir [HS02] reduced the private key size to $O(logn)$. The best known fully collusion system is presented by Boneh, Gentry and Waters [BGW05] which both the size of ciphertexts and public key are $o(\sqrt{n})$.

2.7 Digital Signatures

In public key cryptography, an entity can use his public key to encrypt a message, and the obtain ciphertext can be decrypted to original message using his private key. Public key encryption schemes concentrate on message confidentiality. Digital signature is another fundamental notion in public key cryptography, which is first introduced by Diffie and Hellman [DH76] in 1976. It ensures the authenticity of the signer of a digital message or a digital document and prevents the originator from repudiating the same documents. A valid digital signature gives a recipient reason to believe that the message was created by a legal sender, and that it was not altered in transit, since the signature is associated with the signer's private key. Anyone can verify the signature's validity using signer's public key.

Definition 2.13 *A digital signature scheme typically consists of three algorithms: KeyGen, Sign and Verify.*

- *KenGen: Take a security parameters k as input, the algorithm outputs a private key (sk) and a public key (pk).*
- *Sign: Take the private key sk and a message M as inputs, the algorithm returns a signature σ as output.*
- *Verify: Given (pk, M, σ), either accepts or rejects, such that if $(sk, pk) \leftarrow KeyGen(1^k)$ and $\sigma \leftarrow Sign(sk, M)$, then $Verify(pk, M, \sigma) = accept$.*

Goldwasser, Micali, and Rivest [GMR88] lay out a hierarchy of attack models against digital signatures:

- In a key-only attack, the attacker is only given the public verification key.
- In a known message attack, the attacker is given valid signatures for a variety of messages known by the attacker but not chosen by the attacker.

- In an adaptive chosen message attack, the attacker first learns signatures on arbitrary messages of the attacker's choice.

There are various extensions of digital signatures, such as blind signature [CH91], group signature [CH91, CP95], ring signature [RST01], proxy signature [MUO96] and threshold signature [WLC98].

2.8 Proxy Signature

The concept of proxy signature allows an entity, called original signer, to delegate signing capability to another entity, called proxy signer, so that the proxy signer can sign the document on behalf of the original signer. Proxy signature can not be distinguished from standard signatures. Based on the delegation type, proxy signatures are classified into: full delegation, partial delegation and delegation by warrant [WBZD04]. In full delegation, original signer's private key is given to proxy signer. In a partial delegation scheme, a proxy signer signs a document using a new key, called proxy private key, which is different from the original signer's private key. In a delegation by warrant scheme, the proxy signer is limited to sign certain messages by adding a warrant.

The first proxy signature scheme was proposed by Mambo, Usuda and Okamoto [MUO96]. Since then proxy signatures have enjoyed a considerable amount of interest from the cryptographic research community. Furthermore, various extensions of the basic proxy signature primitives have been conducted. They are threshold proxy signature [KPW97, SLT99, Sun99, Zha97], blind proxy signature [LA03a, ZSNL03], proxy signature with warrant recovery [LA03b], nominative proxy signature [PL01b], one-time proxy signature [KJLK01], proxy-anonymous proxy signature [SW02].

Definition 2.14 *A proxy signature scheme consists of four algorithms [MUO96]: $KenGen$, $DeleGen$, $ProxySign$ and $ProxyVer$.*

- *$KeyGen$: The original signer or the proxy signer runs the key pair generation algorithm to generating the key pair respectively,*

$$(pk, sk) \leftarrow KeyGen(1^k)$$

where pk, sk are the public key and the private key.

- *DeleGen: The original signer runs the delegation generation algorithm which takes a warrant ω, a private key sk_o of the original signer as input and outputs a delegation d,*

$$d \leftarrow DelGen(\omega, sk_o).$$

- *ProxySign: The proxy signer runs the proxy signing algorithm to generate a proxy signature σ by inputting a message m, a warrant ω, a delegation d of the warrant ω and a private key sk_p of the proxy signer*

$$\sigma \leftarrow ProxySign(m, \omega, d, sk_p).$$

- *ProxyVer: The proxy signature verifying algorithm which inputs a message m, a warrant ω, a proxy signature σ, a public key pk_o of the original signer, and a public key pk_p of the proxy signer and outputs true or false, $ProxyVer(m, \omega, \sigma, pk_o, pk_p) = true$, if $\sigma \leftarrow ProxySign(m, \omega, d, sk_p)$,and $d \leftarrow DeleGen(\omega, sk_o)$. Otherwise, $ProxyVer(m, \omega, \sigma, pk_o, pk_p) = false$.*

2.9 Ring Signature

Ring signature was first introduced by Rivest, Shamir and Tauman [RST01] in 2001, which allows users to sign a message on behalf of a ring of legitimate signer, without disclosing the signer's identity. It is a simplified group signature which consists of only users without managers. To produce a ring signature, the actual signer declares an arbitrary set of possible signers including himself, and computes the signature entirely by himself using only his secret key and other's public keys. Many ring signature schemes and its variants have been proposed, such as threshold ring signature [LK07], identity-based ring signature [AL05], and proxy ring signature [AL04].

Definition 2.15 *We call a set of possible signers a ring, the ring members who produces the actual signature the signer and each of the other ring members a non-signer. A ring signature scheme is defined by two procedures:*

- *$ring - sign(m, P_1, P_2, ..., P_r, s, S_s)$ which produces a ring signature σ for the message m, given the public keys $P_1, P_2, ..., P_r$ of the r ring members, together with the secret key S_s of the $s - th$ member (who is the actual signer).*

- *$ring - verify(m, \sigma)$ which accepts a message m and a signature σ (which includes the public keys of all the possible signers), and outputs either true or $false$.*

2.10 Secret Handshake

Balfanz *et al.* introduced a secret handshake scheme in 2003 [BDS+04]. It allows two members from the same group to identify each other secretly. Each party reveals his/her affiliation to other one only if this party is also a group member [CJT04].

Definition 2.16 *A secret handshake scheme includes of the following algorithms [CJT04]:*

- *Setup: Take a security parameter k as input, the algorithm returns the public parameters $params$ to all subsequently generated groups.*
- *CreateGroup: Group Authority (GA) runs the algorithm by taking $params$ as input, and outputs the group public key PK_G and the GA's private key t_{GA}.*
- *AddMember: GA runs the algorithm by taking S_{GA} as input and sharing inputs: $params$, PK_G and the string ID of size of regulated by $params$. The group member will obtain the trapdoor t produced by GA for the above ID.*
- *Handshake: this algorithm is ran between players A, B on input ID_A, ID_B, and $params$. The private input of A is (t_A, PK_A) and the private input of B is (t_B, PK_B). The output for either party is $reject$ or $accept$.*

2.11 Hidden Credentials

The concept of hidden credentials was first proposed by Holt, Bradshaw, Seamons and Orman [HBSO03] in 2003, which let Alice encrypt a message in such a way that Bob can only decrypt it when he possesses the right credentials. That is, his credentials are used as the decryption key.

Definition 2.17 *A hidden credential system consists of two step algorithms ,an encryption function and a decryption function. Before defining the function which set up a credential issuer [HBSO03]:*

- *$CA_create()$: Create a credential authority or issuer.*
- *$CA_issue(nym, attribute)$: Create a credential certifying that the user identified by nym possesses attribute.*

The encryption and decryption functions are as follows:

- *$CT = HC_E(R, nym, P)$: Encrypt a resource R guarded by a policy P with intended recipient identified by nym, and return ciphertext CT.*
- *$R = HC_D(CT, Cred_1, ..., Cred_n)$: Decrypt CT, and return R if $Cred_1, ..., Cred_n$ contains credentials issued with respect to nym which are sufficient to satisfy P.*

2.12 Oblivious Signature-based Envelopes

An Oblivious Signature-based Envelopes scheme (OSBE) enables a sender to encrypt a message and sends the ciphertext to a receiver who can decrypt the it if and only if the receiver has the valid third party's signature on a previously agreed upon messages. However, the sender does not know whether the receiver has the signature or not.

Definition 2.18 *An OSBE scheme includes four entities: CA, send S, receiver R_1, and receiver R_2; and three algorithms [LDB03]:*

- *Setup: CA takes a security parameter k and two messages M_1 and M_2 as input. The CA generates system parameters. As part of this, the key generation algorithm of Sig is executed to create a signing key whose public key is denoted by PK. The CA keeps the secret signing key to itself. It gives the system parameters and the public key PK to S, R_1, and R_2. In addition, the CA gives the message M_1 to S, R_1, and R_2, the message M_2 to S, and the signature $\sigma = Sig_{PK}(M)$ to R_1.*
- *Interaction: The interaction phase has two kinds of interactions, one between S and R_1 and the other between S and R_2.*
- *Open: After an interaction between S and R_1, R_1, the algorithm outputs the message P in the open phase. After an interaction between S and R_2, R_2 does nothing in the open phase.*

An OSBE scheme mustsatisfy three properties:

- *soundness: If the authorized receiver R_1 (who has the signature σ on message m) can output P;*
- *Obliviousness: S does not know whether it is communicate with R_1 or R_2;*
- *Semantic Security Against the Receiver: The receiver if R_2 learn noting about P.*

Oblivious signature-based envelopes scheme was first introduced by Li, Du and Boneh [LDB03], and they presented three concrete OSBE scheme: RSA-OSBE , BLS-OSBE and Rabin-OSBE [NT05].

2.13 Policy-based Cryptography

Bagga and Molva proposed policy-based cryptography firstly in 2006 [BM05], which includes two cryptographic primitives: policy-based encryption and policy-based signature. Policy-based encryption allows a sender to encrypt data according to a set of policies so that only entities satisfying the policies can decrypt the received message. Policy-based signature allows a to generate a digital signature on data with a set of policies so that only the entities who satisfy the policies can generate a valid signature [BM05].

Policy Model: Policy *pol* is fulfilled by a set of credentials generated by one trusted authority. Let TA denote a trusted authority who issues credentials associated with policies. Let A denote an assertion issued by TA. Each assertion A may be a hash value of some statement, such as "Virginblue agency". In general, a policy *pol* can be represented in the disjunctive normal form (DNF) or the conjunctive normal form (CNF). For example, a policy in DNF is as following: $pol = ((A_{1,1,1} \wedge A_{1,1,2}) \vee (A_{1,2,1} \wedge A_{1,2,2}) \vee A_{1,3,1})) \wedge ((A_{2,1,1} \wedge A_{2,1,2}) \vee A_{2,2,1} \vee (A_{2,3,1} \wedge A_{2,3,2}))$. In order to address these two normal forms, a policy denoted *pol* will be written in conjunctive-disjunctive normal form (CDNF):

$$pol = \wedge_{i=1}^{b}[\vee_{j=1}^{b_i}[\wedge_{k=1}^{b_{i,j}} < A_{i,j,k} >]]$$

where $1 \leq i \leq b, 1 \leq j \leq b_i, 1 \leq k \leq b_{i,j}$. Thus, in CNF form, policies can be expressed as $b_{i,j} = 1$ for all i, j, while in DNF form, policies can be expressed as $b = 1$.

Definition 2.19 *A policy-based encryption scheme includes two algorithms: PolEnc and PolDec [BM05].*

- *PolEnc: A sender takes a message m and policy Pol as input, the algorithm outputs a ciphertext C where presents the message m is encrypted according to policy Pol.*

$$c = PolEnc(m, Pol)$$

- *PolDec: A receiver takes ciphertext C, policy Pol and a set of credentials $Cred_{j_1,\ldots,j_b}$ as input, the algorithm outputs a message m.*

$$PolDec(c, Pol, Cred_{j_1,\ldots,j_b}) = m$$

Definition 2.20 *A policy-based signature scheme includes two algorithms: PolSig and PolVer [BM05].*

- *PolSig: A signer takes a message m, policy Pol and a set of credentials $Cred_{j_1,\ldots,j_b}$ as input, the algorithm outputs a signature σ where presents the message m is signed according to policy Pol.*

$$\sigma = PolSig(m, Pol, Cred_{j_1,\ldots,j_b})$$

- *PolVer: A receiver takes a signature σ, policy Pol and a message m as input, the algorithm outputs $true$ if σ is a valid signature on m according to policy Pol. Otherwise, it returns $false$.*

$$PolVer(m, Pol, Cred_{j_1,\ldots,j_b}) = true$$

2.14 Attribute-based Cryptography

Attribute-based cryptography is considered as a promising primitive for digital signature, which includes attribute-based encryption and attribute-based signature firstly proposed by Sahai and Waters [SW05].

An essential feature of ABE schemes is the property of collusion resistance, which guarantees that a ciphertext can leak no information about the plaintext to users whose attributes do not satisfy the considered policy, even if the union of the attributes of these colluding users satisfies the policy. This property is essential to

guarantee a reasonable level of security in many of the applications of ABE schemes, like anonymous access control or access to encrypted data [HLR10].

A notion similar to ABE but without collusion resistance property has been considered under different names: policy-based encryption [BM06]. In most of current ABE paper [NAM05, LCLS07, HLR10, GPSW06], the process method of attributes is based on threshold, however, boolean expression written in generic conjunctive-disjunctive normal form is used to process complex policy-attributes in policy-based cryptography papers.

Chapter 3

Policy-Based Authentication for Mobile Agents

3.1 Introduction

Mobile agents have wide applications in electronic commerce. Mobile agents have the ability of migrating across different execution environments [KBC00, LKK01, KC02, Hoh98]. For example, a customer releases a mobile agent to search a suitable flight ticket. The mobile agent may travel around the network and negotiate with the suitable sellers. After a deal has been made, the contract will be sent to the customer. Due to their mobility and autonomy, mobile agents can provide an efficient, convenient, and robust framework for distributed e-commerce applications.

Although mobile agent is a promising tool for e-commerce, it also encounters some serious security threats. When a mobile agent is executed on a host, it has an asymmetric relation with the host. The host can access the agent's code, data and state; therefore, it is infeasible for an agent to carry any secret information without exposing it to malicious hosts [Che98, ACCK01]. In a hostile environment, the following security problems should be taken into account [ST98b, LKK01, ZMZ09]:

- Code and execution integrity: Can a mobile agent protect itself against tampering by a malicious host?
- Code privacy: Can a mobile agent conceal the program it wants to execute?
- Computing with secrets in public: Can a mobile agent remotely sign a document without disclosing the private key?
- Non-repudiations: Can a mobile agent provide the fairness of contract? (Both the entities can not repudiate the contract they signed before.)

There exist several solutions to these problems. Sander and Tschudin [ST98b] proposed a scheme to solve the problems of code integrity, code privacy and computation without disclosing the private key. In their scheme, the concept of computing with an encrypted function (CEF) was introduced to conceal the signature function by composing it with the encryption function. However, the secure implementation was not provided. Kotzanikolaou, Burmester and Chrissikopoulos [KBC00] implemented CEF using an RSA-based undetachable signature scheme. Customer's private key is hidden in the encrypted signature function and its secrecy is based on the RSA assumption. Generally, complexity problems, such as, integer factorization problem and discrete logarithm problem, can be used to conceal customer's private key and achieve the integrity of mobile code. However, their schemes did not address the fairness of the contract. Lee, Kim and Kim [LKK03] proposed a scheme to solve the repudiation problem. The proxy signature was used to allow all the hosts to generate a valid signature after executing the mobile agent [KBC00, SL03, PL01a, LKPY06, LKK01, KJLK01]. Although it ensures the fairness of the contract, the customer receives the service from all the hosts in the agent network including the malicious ones. Recently, Zhang, Mu and Zhang [ZMZ09] proposed a secure mobile agent scheme to achieve host authentication with designated hosts, where only selected hosts can be included in the agent network. They added the ID information in the designated host list that allows certain hosts to perform an agent task. Therefore, the repudiation problem can be solved.

Policy-based schemes can address the problem of host authentication with designated host [ZMZ09]. A policy specifies the constrains under which a specific action can be performed on a certain sensitive resource [BM06, BM05, TSM09b, TSM09a]. The Zhang, Mu and Zhang scheme [ZMZ09] can be regarded as a simple policy-based scheme. There is only one policy, that is, the ID information in their authentication system. However, it is not sufficient to meet the requirements in the real-world applications. For example, in the case of buying a flight ticket, a customer may care about not only the kind of host, but also the location and reputation of the host. Therefore, the host must satisfy a set of policies. This raises a challenging question: How to authorize designated hosts that satisfy a set of policies and can generate a valid policy-based signature in a hostile environment?

In this section, we provide a sound answer to the question by introducing a novel scheme, which allows designated hosts to perform an agent task. The list of designated hosts can be chosen by the agent owner according to the set of policies.

In the proposed scheme, hosts are treated as proxy signers, where Bagga and Molva signature scheme [BM05] was used as the basis to allow only the hosts who have the proper policy set to generate a valid "proxy" signatures. In our scheme, we require each mobile agent to hold a set of credentials, which is based on the short signature scheme defined in [BLS04].

We also define a rigorous security model for mobile agent transaction, which captures the most powerful attacks including adaptive chosen message and adaptive chosen host attacks. The security of our scheme is based on the hardness of Computational Diffie-Hellman problem in the random oracle model.

3.2 Mobile Agent Model

3.2.1 Policy Model

Policy *pol* is fulfilled by a set of credentials generated by one trusted authority. In order to ensure a host to be authorized to perform a special action $sign$ on a sensitive resource res, he has to prove his compliance to the policy defined by the agent owner. He has to prove that he possesses a minimal set of credentials that is required by *pol* to permit action $sign$ on res. Let TA denote a trusted authority who issues credentials associated with policies. Let A denote an assertion issued by TA. Each assertion A may be a hash value of some statement, such as "Virginblue agency". In general, a policy *pol* can be represented in the disjunctive normal form (DNF) or the conjunctive normal form (CNF). For example, a policy in DNF is as following: $pol = ((A_{1,1,1} \wedge A_{1,1,2}) \vee (A_{1,2,1} \wedge A_{1,2,2}) \vee A_{1,3,1})) \wedge ((A_{2,1,1} \wedge A_{2,1,2}) \vee A_{2,2,1} \vee (A_{2,3,1} \wedge A_{2,3,2}))$. In order to address these two normal forms, a policy denoted *pol* will be written in conjunctive-disjunctive normal form (CDNF):

$$pol = \wedge_{i=1}^{b}[\vee_{j=1}^{b_i}[\wedge_{k=1}^{b_{i,j}} < A_{i,j,k} >]]$$

where $1 \leq i \leq b, 1 \leq j \leq b_i, 1 \leq k \leq b_{i,j}$. Thus, policies expressed in CNF form are such that $b_{i,j} = 1$ for all i, j, while policies expressed in DNF form are such that $b = 1$. Here, b denotes the number of conjunctive operations, i, j, k are used as subscripts of an assertion.

3.2.2 Mobile Agent Model

The major procedures of our mobile agent system for executing a task in online applications consist of following phases: Customer Setup, Agent Setup, TA Setup, Agent Dispatch, Host Execution and Verification.

- **Customer Setup**: The customer decides the services he intends to receive and selects a set of hosts for inclusion.
- **Agent Setup**: The customer generates a delegation token based on its requirement and embeds it in the mobile agent. This token includes the list of designated hosts that are permitted in the agent network.
- **Agent Dispatch**: The mobile agent travels in the network and searches for host from the list.
- **Host Execution**: When a mobile agent arrives at a host, the host checks the validity of the delegation token. If it is invalid, the host will stop execution; otherwise, it will execute the mobile agent following the designated procedure and generate the signature.
- **Verification**: Anyone can verify whether the signed service is valid, following the verification algorithm.

Figure 3.1 demonstrates an example of a mobile agent executing a task on behalf of its owner in a general online application. The customer defines a set of policies, that also means the customer selects a set of hosts (1 and 3). The policy set is $pol = pol_1 \wedge pol_2 \wedge pol_3$. In pol_1, it consists of three sub-policy, that is, $pol_1 = pol_{1,1} \vee pol_{1,2} \vee pol_{1,3}$. It means the host must hold at least one sub-policy from pol_1, pol_2 and pol_3, respectively. The customer then generates a delegation token based on the task and the designated host list, and embeds the token in a mobile agent. Mobile agent then travels in the network searching for the designated host from the list. When an agent arrives at a host, say Host 1, Host 1 verifies the delegation token prior to an execution. Then Host 1 can execute the mobile agent because it holds the minimal set of policies that is required according to the defined policy set by agent owner. Host 2 is excluded from the agent network, because his policy does not satisfy all the policies defined in the delegation token.

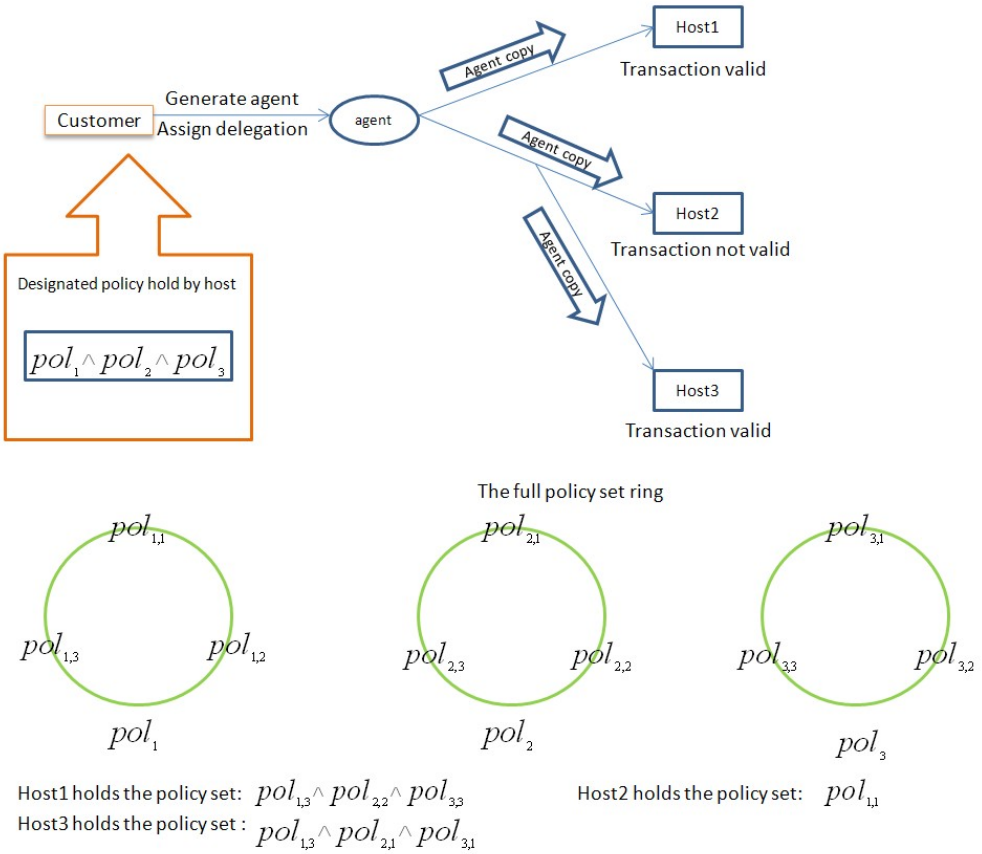

Figure 3.1: Mobile agent model

3.3 Definition and Security Model

3.3.1 Notation

pol: a set of policies held by a customer or a host.

TA: a trusted authority who issues credentials associated to the policy.

A: an assertion issued by TA, which is a statement of hash value of a policy.

Cre: a credential for an assertion issued by TA and signed by TA's secret key.

D: delegation token, signed by a customer's private key.

sk_C: a private key of a customer.

pk_C: a public key of a host.

H: a random hash function.

3.3.2 Definition

There are four parties involved in our scheme: Customer, Agent, Host and Third Party (TA). A Policy-based Authentication scheme for mobile agent system can be described by following phases:

- **System Setup**: This algorithm takes a security parameter 1^{k_S} as input, and returns systems parameters *params*.
- **Customer KeyGen**: The algorithm takes *params* and a security parameter 1^{k_C} as input, returns the customer private/public key pair (sk_C, pk_C). It is run by a customer in the system.
- **TA KeyGen**: The algorithm takes *params* and a security parameter 1^{k_T} as input, returns the TA private/public key pair (sk_{TA}, pk_{TA}). It is run by TA.
- **Host KeyGen**: This algorithm takes *params* and a security parameter 1^{k_H} as input, returns the host private/public key pair (sk_H, pk_H). It is run by a host in the system.
- **CreGen**: This algorithm takes *params*, an assertion A and TA private key sk_{TA} as input, returns the credential Cre of an assertion A for a user in the system. It is run by TA.
- **DeleGen**: This algorithm takes *params*, the private key of a customer sk_C, the customer's warrant $\omega \in \{0,1\}^*$ and a policy *pol* defined by a customer as input, then returns the delegation token D to a customer in the system.
- **Agent Dispatch**: The customer embeds D in the mobile agent. Then the mobile agent travels around the network and searches for host from the designated host list.
- **Host Execution**: This algorithm takes *params*, a message m, the delegation token D, the policy *pol*, the set of credentials $Cre_{j_1,j_2,\ldots,j_b}$ and the host private key sk_H as input, and returns a signature σ signed by the host.
- **Verifying**: This algorithm takes *params*, the message/signature pair (m, σ), the warrant ω, the customer public key pk_C, the host public key pk_H and the full policy *pol* as input, returns Valid if σ is a signature on m of (ω, D) signed with sk_C and credentials $Cre_{j_1,j_2,\ldots,j_b}$ corresponds to the full policy. Otherwise, returns invalid.

3.3.3 Security Models

We classify the adversary into four types of attacks as follows:

- Adversary, who possesses a set of credentials $Cre_{j_1,j_2,\ldots,j_b}$ and the private key sk_H of any host, but does not have the delegation token D, tries to forge a valid signature. We can consider this adversary as a malicious host in the designated list.

- Adversary, who possesses the delegation token D and the private key sk_H of any host, but does not possess the set of satisfied credentials $Cre_{j_1,j_2,\ldots,j_b}$, tries to forge a valid signature. We can consider this adversary as a malicious host that is not in the designated list.

- Adversary, who possesses the delegation token D and the set of credentials $Cre_{j_1,j_2,\ldots,j_b}$, but does not have the private key sk_H, of any host, tries to forge a valid signature. We can consider this adversary as a malicious customer.

- Adversary, who can easily get the set of credentials $Cre_{j_1,j_2,\ldots,j_b}$ and the private key sk_H of a host, tries to forge a valid signature signed by sk_H without D. We can consider this adversary as a malicious TA.

Adversary one and Adversary four both can easily get the set of credentials $Cre_{j_1,j_2,\ldots,j_b}$ and the private key sk_H of a host, they can be seen as one situation. Therefore, we propose security models and divide the adversaries into three types:

1. $\mathcal{A}_1$ can forge a valid signature without delegation token D.

2. $\mathcal{A}_2$ can forge a valid signature without the credentials $Cre_{j_1,j_2,\ldots,j_b}$.

3. $\mathcal{A}_3$ can forge a valid signature without the private key sk_H of a host.

Existential Unforgeability Against $\mathcal{A}_1$

We assume that $\mathcal{A}_1$ possesses the key pair of any host they chosen and the credentials satisfied the full policy, but does not possess the delegation token D. It is defined using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_1$:
Setup:

- $\mathcal{C}$ runs the **System Setup** algorithm to obtain system parameters *Params*.

- $\mathcal{C}$ runs the **Customer KeyGen** algorithm to obtain the key pair of a customer (sk_C, pk_C) and sends pk_C to $\mathcal{A}_1$.
- $\mathcal{C}$ runs the **TA KeyGen** algorithm to obtain the key pair of TA (sk_{TA}, pk_{TA}) and sends TA public key pk_{TA} to $\mathcal{A}_1$.
- $\mathcal{C}$ runs the **Host KeyGen** algorithm to obtain the key pair of a host (sk_H, pk_H) and sends the key pair to $\mathcal{A}_1$.

Delegation Queries: $\mathcal{A}_1$ can choose the warrant ω adaptively and the designated host list $X = \{H(A_{1,1}), ..., H(A_{1,j_1}), H(A_{2,1}), ..., H(A_{2,j_2}), ..., H(A_{i,1}), ..., H(A_{i,j_i})\}$, then submits (ω, X) to $\mathcal{C}$. In response, $\mathcal{C}$ runs the algorithm **DeleGen** (ω, X) and returns D to $\mathcal{A}_1$, where D is the delegation token of combination of X and the warrant ω chosen by $\mathcal{A}_1$.
Forgery: Finally, $\mathcal{A}_1$ can output a signature $\sigma^*(m^*, \omega^*, X^*, \sigma^*)$ on messages m^* and win the game if:

1. (ω^*, X^*) has not been requested before.
2. $Verify(params, pk_C, pk_H, pk_{TA}, m^*, \omega^*, X^*, \sigma^*)$=valid.

We define $Suc_{\mathcal{A}_1}$ to be the probability that the adversary $\mathcal{A}_1$ wins the above game.

Definition 3.1 *The scheme is (t, q_D, ϵ)-secure against $\mathcal{A}_1$ attack if there is no adversary $\mathcal{A}_1$ who can win this game in time at most t, and make at most q_D delegation queries, $Suc_{\mathcal{A}_1}$ is at least ϵ.*

Existential Unforgeability Against $\mathcal{A}_2$

$\mathcal{A}_2$ does not have the credentials corresponding to the policy which is defined by a customer, but does possess the delegation token D and the key pair of any host. It is defined using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_2$:
Setup:

- $\mathcal{C}$ runs the **System Setup** algorithm to obtain system parameters *Params*.
- $\mathcal{C}$ runs the **TA KeyGen** algorithm to obtain the key pair of TA (sk_{TA}, pk_{TA}), and sends pk_{TA} to $\mathcal{A}_2$.
- $\mathcal{C}$ runs the **Customer KeyGen** algorithm to obtain the key pair of a customer (sk_C, pk_C), and sends the customer public key pk_C to $\mathcal{A}_2$.

- $\mathcal{C}$ runs the **Host KeyGen** algorithm to obtain the key pair of TA (sk_H, pk_H), and sends the key pair to $\mathcal{A}_2$.

Credential Queries: $\mathcal{A}_2$ chooses an assertion $A_{i,j,k}$ and submits them to $\mathcal{C}$. In response, $\mathcal{C}$ runs algorithm **CreGen**$(params, A_{i,j,k})$ and returns the credential $Cre_{i,j,k}$ to $\mathcal{A}_2$.

Forgery: $\mathcal{A}_2$ can output a signature $\sigma*(m*, \omega*, X*, A_{i,j}*, Cre^*_{j_1,j_2,\ldots,j_b}, \sigma*)$ on messages $m*$ and wins the game if:

1. $A_{i,j}{}^*$ is a set of assertions which are owned by a host, and have not been queried before.

2. $Verify(params, pk_C, pk_H, pk_{TA}, m^*, \omega^*, X^*, Cre^*_{j_1,j_2,\ldots,j_b}, \sigma^*) =$ valid.

We define $Suc_{\mathcal{A}_2}$ to be the probability that the adversary $\mathcal{A}_2$ wins the above game.

Definition 3.2 *The scheme is (t, q_C, ϵ)-secure against $\mathcal{A}_2$ attack if there is no adversary $\mathcal{A}_2$ who can win this game in time at most t, and make at most q_C credentials queries, $Suc_{\mathcal{A}_2}$ is at least ϵ.*

Existential Unforgeability Against $\mathcal{A}_3$

We assume that $\mathcal{A}_3$ possesses the set of credentials and the delegation token D, but does not have the access to the private key of a host. It is defined using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_3$:

Setup:

- $\mathcal{C}$ runs the **System Setup** algorithm to obtain system parameters *Params*.

- $\mathcal{C}$ runs the **Host KeyGen** algorithm to obtain the key pair of a host (sk_H, pk_H), and returns the host public key pk_H to $\mathcal{A}_3$.

- $\mathcal{C}$ runs the **TA KeyGen** algorithm to obtain the key pair of TA (sk_{TA}, pk_{TA}), and returns the TA public key pk_{TA} to $\mathcal{A}_3$.

- $\mathcal{C}$ runs the **Customer KeyGen** algorithm to obtain the key pair of a customer (sk_C, pk_C), and returns the key pair to $\mathcal{A}_3$.

Signature Queries: Adversary $\mathcal{A}_3$ can make any signature query of $(m, \omega, X, Cre_{j_1,j_2,\ldots,j_b})$. $\mathcal{C}$ runs the algorithm to get σ and returns it to $\mathcal{A}_3$.

Forgery: $\mathcal{A}_3$ outputs a signature σ^* $(m^*, \omega^*, X^*, \sigma^*)$ on messages m^* and wins the game if:

1. σ^* has not been requested before.

2. $Verify(param, pk_C, pk_H, pk_{TA}, m^*, \omega^*, X^*, \sigma^*) =$ valid.

We define $Suc_{\mathcal{A}_3}$ to be the probability that the adversary $\mathcal{A}_3$ wins the above game.

Definition 3.3 *The scheme is (t, q_s, ϵ)-secure against $\mathcal{A}_3$ attack if there is no adversary $\mathcal{A}_3$ who can win this game in time at most t, and make at most q_s signature queries, $Suc_{\mathcal{A}_3}$ is at least ϵ.*

3.4 Policy-based Authentication for Mobile Agent Scheme

System Setup: Run the algorithm as input 1^k to generate $paramas = (q, \mathbb{G}_1, \mathbb{G}_2, e)$. Randomly pick a generator $P \in \mathbb{G}_1$. Define five hash functions: $H, H_0 : \{0,1\}^* \to Z_q^*; H_1, H_2, H_3 : \{0,1\}^* \to \mathbb{G}_1$.
Customer KeyGen: Customer selects a random number $s_C \in Z_q^*$ and sets the customer key pair $(pk_C, sk_C) = (s_C P, s_C)$.
Host KeyGen: Host selects a random number $s_H \in Z_q^*$ and sets the host key pair $(pk_H, sk_H) = (s_H P, s_H)$.
TA KeyGen: TA randomly picks at a master key $s_{TA} \in Z_q^*$ and sets the TA key pair $(pk_{TA}, sk_{TA}) = (s_{TA} P, s_{TA})$.
CreGen: Given a valid assertion A and TA private key s_{TA}, this algorithm outputs the credential $C(pk_{TA}, A) = s_{TA} H_2(A) \in \mathbb{G}_1$.
DeleGen: Customer builds the designated host list and makes the delegation token as following:

- Define a set of policies: $pol_C = \wedge_{i=1}^{b}[\vee_{j=1}^{b_i}[\wedge_{k=1}^{b_{i,j}} < A_{i,j,k} >]]$;
- Compute $H(A_{i,j}) = \sum_{k=1}^{b_{i,j}} H(A_{i,j,k}), 0 \le i \le b, 0 \le j \le b_i$;
- Set $X = \{H(A_{1,1}), H(A_{1,2}), H(A_{1,3}), ..., H(A_{1,b_1}), H(A_{2,1}), ..., H(A_{2,b_2}), ..., H(A_{i,1}), H(A_{b,b_i})\}$;
- Compute
$$V = (\sum_{i=1}^{b} \sum_{j=1}^{b_i} H(A_{i,j}))P \in \mathbb{G}_1,$$

where V is a value representing the designated host list.

- Compute the delegation token

$$D = s_C H_1(\omega, V) \in \mathbb{G}_1,$$

where ω is the warrant.

- Embed (ω, X, D, pol_C) in the mobile agent.

Host execution: Upon receiving (ω, X, D, pol_C) from the agent, the host carries out the following:

- Verify the delegation token D by checking

$$e(D, P) \stackrel{?}{=} e\Big(H_1(\omega, V), pk_C\Big).$$

- Compute $H(A_{i,j_i})$, where A_{i,j_i} are the set of policies the host holds. If $H(A_{i,j_i}) \notin$ X, the host stops the execution. Otherwise, given a message m, pol_C, and the set of credentials $Cre_{j_1,...,j_b(pol_H)}$, we divide the full policy pol_C into b blocks, doing as following in any one of the blocks, for $i = 1, ..., b$:

  1. The host, who satisfies the policy, picks randomly $Y_i \in \mathbb{G}_1$,and then computes

$$X_{i,j_i+1} = e(P, Y_i);$$

  2. For $l = 1, 2, ..., j_i - 1, j_i + 1, ..., b_i$,

     (a) Compute

$$\tau_{i,l} = \prod_{k=1}^{b_{i,j}} e\Big(pk_{TA}, H_2(A_{i,j,k})\Big) = \prod_{k=1}^{b_{i,j}} e\Big(P, s_{TA} H_2(A_{i,j,k})\Big);$$

     (b) Pick randomly $Y_{i,l} \in \mathbb{G}_1$, then compute

$$X_{i,l+1} = e(P, Y_{i,l}) \cdot \tau_{i,l}^{H_0(m||X_{i,l}||pol_C)} \cdot e\Big(P, D + s_H H_3(m||\omega||pol_C)\Big).$$

  3. On solving the ring equation the host obtains

$$Y_{i,j_i} = Y_i - H_0(m||X_{i,l}||pol_C) \cdot \sum_{k=1}^{b_{i,j_i}} C_{i,j_i} D - s_H H_3(m||\omega||pol_C).$$

- Compute

$$Y = \sum_{i=1}^{b} \sum_{j=1}^{b_i} Y_{i,j}.$$

- Set the signature to be

$$\sigma = ([X_{i,1}, X_{i,2}, ..., X_{i,b_i}]_{1 \leq i \leq b}, Y).$$

Verification: Given a message m, the full policy pol_C, the signature $\sigma = \Big([X_{i,1}, X_{i,2}, ..., X_{i,b_i}]_{1 \leq i \leq i}, Y, pk_C, pk_H, pk_{TA}\Big)$, the customer checks the signature as follows:

- Compute

$$Z_1 = \prod_{i=1}^{b} \prod_{j=1}^{b_i} X_{i,j};$$

- For $i = 1, ..., b$, and for $j = 1, ..., b_i$, compute

$$\tau_{i,j} = \prod_{k=1}^{b_{i,j}} e\Big(pk_{TA}, H_2(A_{i,j,k})\Big);$$

- Compute

$$Z_2 = e(P, Y) \cdot \prod_{i=1}^{b} \prod_{j=1}^{b_i} \tau_{i,j}^{H_0(m||X_{i,l}||pol_C)} \cdot e\Big(H_1(\omega, V), pk_C\Big) \cdot e\Big(H_3(m||\omega||pol_C), pk_H\Big);$$

- If $Z_1 = Z_2$, the signature is valid. Otherwise, it is invalid.

3.5 Security Analysis

3.5.1 Existential Unforgeability Against $\mathcal{A}_1$

Theorem 3.1 *If there exists an adversary $\mathcal{A}_1$ who can $(t, q_H{}^*, q_D, \epsilon)$-break the proposed scheme then there exists another algorithm $\mathcal{B}$ that can use $\mathcal{A}_1$ to solve an instance of the CDH problem in $\mathbb{G}_1$.*

Proof: Algorithm $\mathcal{B}$ is given a random instance (P, aP, bP) of the CDH problem in $\mathbb{G}_1$. Its goal is to compute abP by interacting with adversary $\mathcal{A}_1$ as described below. The hash functions H_1 is regarded as the random oracles during the proof.
Setup:

- Run the **System Set** algorithm to obtain the system's parameters *params*.
- Set $pk_C = aP$, where aP is one of input of CDH problem.
- Maintain H_1-*list* that stores the results of queries to random oracles. Initially, it is empty.
- Return *params* and pk_C to $\mathcal{A}_1$.

H_1 **Queries**: Adversary $\mathcal{A}_1$ can query the result of H_1 as inputs (ω, V) at any time. $\mathcal{B}$ checks the H_1-*list* first:

- If there exists an item $((\omega, V), h_1, r, coin)$ in the H_1-*list*, $\mathcal{B}$ will return h_1 to $\mathcal{A}_1$.
- Otherwise, $\mathcal{B}$ tosses a coin $coin \in \{0, 1\}$ such that, $Pr[coin] = \delta$
  - If $coin = 1$, $\mathcal{B}$ chooses $r \in_R Z_p^*$ and computes $h_1 = rbP$, where bP is one of the instances of CDH problem.
  - Otherwise, $coin = 0$, $\mathcal{B}$ chooses $r \in_R Z_p^*$ and computes $h_1 = rP$.

Delegation Queries: We assume that all of the tuple $((\omega, V), h_1, r, coin)$ have been submitted. $\mathcal{B}$ can query (ω, V) if the tuple does not exist.

- If $coin = 0$, then $H_1(\omega, V) = rP$. $\mathcal{B}$ can compute $D = r \cdot aP = a \cdot rP = s_C H_1(\omega, V)$.
- If $coin = 1$, $\mathcal{B}$ terminates the simulation and reports failure.

Forgery: $\mathcal{A}_1$ will output a valid signature tuple $(m^*, \omega^*, X^*, \sigma^*)$.

- If $coin$=0, $\mathcal{B}$ aborts.
- If $coin$=1, this step is to apply the general forking lemma. $\mathcal{B}$ then replays $\mathcal{A}_1$ with the same tuple but different H_1. Suppose H_1 outputs h_1 and $h_1^{'}$ in the first round and second round, respectively.

$$\begin{cases} \sigma^* = \Big([X_{i,1}^*, X_{i,2}^*, ..., X_{i,b_i}^*]_{1 \le i \le b}, Y^*\Big), \\ \sigma^{'*} = \Big([X_{i,1}^{'*}, X_{i,2}^{'*}, ..., X_{i,b_i}^{'*}]_{1 \le i \le b}, Y^{'*}\Big). \end{cases}$$

Where,

$$\begin{cases} Y^* = \Big((Y_1^* - h_{01}^* \cdot \sum_{k=1}^{b_i,j_1} C_{1,j_1}^* - ar^*bP - s_H \cdot h_3^*) + \\ \quad (Y_2^* - h_{02}^* \cdot \sum_{k=1}^{b_i,j_2} C_{2,j_2}^* - ar^*bP - s_H \cdot h_3^* + ... + \\ \quad (Y_i^* - h_{0i}^* \cdot \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}^* - ar^*bP - s_H \cdot h_3^*)\Big) + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*, \\ Y^{*'} = \Big((Y_1^* - h_{01}^* \cdot \sum_{k=1}^{b_i,j_1} C_{1,j_1}^* - ar^{*'}bP - s_H \cdot h_3^*) + \\ \quad (Y_2^* - h_{02}^* \cdot \sum_{k=1}^{b_i,j_2} C_{2,j_2}^* - ar^{*'}bP - s_H \cdot h_3^* + ... + \\ \quad (Y_i^* - h_{0i}^* \cdot \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}^* - ar^{*'}bP - s_H h_3^*)\Big) + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*. \end{cases} \tag{3.1}$$

$$\begin{cases} Y^* = (Y_1^* + Y_2^* + ... + Y_i^*) - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} C_{1,j_1} + h_{02}^* \\ \quad \sum_{k=1}^{b_{2,j_2}} C_{2,j_2} + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}) - i \cdot ar^*bP - i \cdot \\ \quad s_H h_3^* + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*, \qquad (1) \\ Y^{'*} = (Y_1^* + Y_2^* + ... + Y_i^*) - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} C_{1,j_1} + h_{02}^* \\ \quad \sum_{k=1}^{b_{2,j_2}} C_{2,j_2} + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}) - i \cdot ar^{'*}bP - i \cdot \\ \quad s_H h_3^* + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*. \qquad (2) \end{cases}$$

$$\begin{cases} i \cdot ar^* bP = \sum_{i=1}^{b} Y_i^* - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} C_{1,j_1} + h_{02}^* \sum_{k=1}^{b_{2,j_2}} C_{2,j_2} \\ \qquad + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}) - Y^* - i \cdot s_H h_3^* + \\ \qquad \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*, \\ i \cdot ar^{*'} bP = \sum_{i=1}^{b} Y_i^* - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} C_{1,j_1} + h_{02}^* \sum_{k=1}^{b_{2,j_2}} C_{2,j_2} \\ \qquad + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}) - Y^{*'} - i \cdot s_H h_3^* + \\ \qquad \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*. \end{cases}$$

Let (1) − (2), we will get

$$abP = \frac{Y^* - Y'^*}{i(r'^* - r^*)}.$$

Therefore, $\mathcal{B}$ can successfully solve the given instance of the CDH problem in $\mathbb{G}_1$.

3.5.2 Existential Unforgeability Against $\mathcal{A}_2$

Theorem 3.2 *If there exists an adversary $\mathcal{A}_2$ who can $(t, q_H{}^*, q_C, \epsilon)$-break the proposed scheme then there exists another algorithm $\mathcal{B}$ that can use $\mathcal{A}_2$ to solve an instance of the CDH problem in $\mathbb{G}_1$.*

Proof: Algorithm $\mathcal{B}$ is given a random instance (P, aP, bP) of the CDH problem in $\mathbb{G}_1$. Its goal is to compute abP by interacting with adversary $\mathcal{A}_2$ as described below. The hash function H_2 is regarded as the random oracles during the proof.
Setup:

- Run the **System Set** algorithm to obtain the system's parameters *params*.
- Set $pk_{TA} = aP$, where aP is one of input of CDH problem.
- Maintain H_2-*list* that stores the results of queries to random oracles. Initially, it is empty.
- Return *params* and pk_{TA} to $\mathcal{A}_2$.

H_2 **Queries**: Adversary $\mathcal{A}_2$ can query the result of H_2 as input an assertion $A_{i,j,k}$ at any time, $\mathcal{B}$ checks the H_2-*list* first:

- If there exists an item $(A_{i,j,k}, h_2, r, coin)$ in the list, $\mathcal{B}$ will return h_2 to $\mathcal{A}_2$.
- Otherwise, $\mathcal{B}$ tosses a coin $coin \in \{0, 1\}$ such that $Pr[coin] = \delta$
  - If $coin = 1$, $\mathcal{B}$ chooses $r \in_R Z_P^*$ and computes $h_2 = rbP$.
  - Otherwise, $coin = 0$, $\mathcal{B}$ chooses $r \in_R Z_P^*$ and computes $h_2 = rP$.

H_3 **Queries**: As in the proof of Theorem 1, $\mathcal{A}_2$ requests (m_i, ω_d, pol_C), $\mathcal{B}$ returns h_3 and adds $((m_i, \omega_d, pol_C), h_3)$ to the H_3-*list* if there is no such entry in the list.
Credentials Queries: $\mathcal{A}_2$ can ask at most q_C credentials queries of $A_{i,j,k}$ chosen by itself. We assume that there is a tuple $(A_{i,j,k}, h_2, r, coin)$ in the H_2-*list* containing $A_{i,j,k}$. $\mathcal{B}$ can make an H_2 query $A_{i,j,k}$ if that tuple does not exist.

- If $coin = 0$, then $H_2(A_{i,j,k}) = rP$. $\mathcal{B}$ can compute $C = aH_2(A_{i,j,k}) = a \cdot rP = r \cdot aP$.
- If If $coin = 1$, $\mathcal{B}$ terminates the simulation and reports failure.

Forgery: $\mathcal{A}_2$ will output a valid signature tuple $(m^*, \omega^*, X^*, Cre_{j_1,...,j_b}{}^*, \sigma^*)$.

- If $coin$=0, $\mathcal{B}$ aborts.
- If $coin$=1, this step is to apply the general forking lemma. $\mathcal{B}$ then replays $\mathcal{A}_2$ with the same tuple but different H_2. Suppose H_2 outputs h_2 and $h_2^{'}$ in the first round and second round, respectively. Here,

$$\begin{cases} \sigma^* = \Big([X_{i,1}^*, X_{i,2}^*, ..., X_{i,b_i}^*]_{1 \le i \le b}, Y^*\Big), \\ \sigma^{'*} = \Big([X_{i,1}^{'*}, X_{i,2}^{'*}, ..., X_{i,b_i}^{'*}]_{1 \le i \le b}, Y^{'*}\Big). \end{cases}$$

where,

$$\begin{cases} Y^* = \Big((Y_1^* - h_{01}^* \cdot \sum_{k=1}^{b_{i,j_1}} a \cdot r_{1,j_1}^* bP - D^* - s_H \cdot h_3^*) + (Y_2^* - \\ \quad h_{02}^* \cdot \sum_{k=1}^{b_{i,j_2}} a \cdot r_{2,j_2}^* bP - D^* - s_H \cdot h_3^*) + ... + (Y_i^* - h_{0i}^* \cdot \\ \quad \sum_{k=1}^{b_{i,j_i}} a \cdot r_{i,j_i}^* bP - D^* - s_H h_3^*)\Big) + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*, \\ Y^{*'} = \Big((Y_1^* - h_0^* \cdot \sum_{k=1}^{b_{i,j_1}} a \cdot r_{1,j_1}^{*'} bP - D^* - s_H \cdot h_3^*) + (Y_2^* - h_{02}^* \\ \quad \cdot \sum_{k=1}^{b_{i,j_2}} a \cdot r_{2,j_2}^{*'} bP - D^* - s_H \cdot h_3^*) + ... + (Y_i^* - h_{0i}^* \cdot \sum_{k=1}^{b_{i,j_i}} a \cdot \\ \quad r_{i,j_i}^{*'} bP - D^* - s_H h_3^*)\Big) + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*. \end{cases} \tag{3.2}$$

$$\begin{cases} Y^* = (Y_1^* + Y_2^* + ... + Y_i^*) - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} r_{1,j_1}^* + h_{02}^* \sum_{k=1}^{b_{2,j_2}} r_{2,j_2}^* + \\ \quad ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} r_{i,j_i}^*) \cdot abP - i \cdot D^* - i \cdot sk_H h_3^* + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*, & (3) \\ Y'^* = (Y_1^* + Y_2^* + ... + Y_i^*) - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} r_{1,j_1}'^* + h_{02}^* \sum_{k=1}^{b_{2,j_2}} r_{2,j_2}'^* + \\ \quad ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} r_{i,j_i}'^*) \cdot abP - i \cdot D^* - i \cdot sk_H h_3^* + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*. & (4) \end{cases}$$

Let

$$\begin{cases} \alpha^* = (h_{01}^* \sum_{k=1}^{b_{1,j_1}} r_{1,j_1}^* + h_{02}^* \sum_{k=1}^{b_{2,j_2}} r_{2,j_2}^* + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} r_{i,j_i}^*) \\ \quad \cdot ar^* bP, \\ \alpha^{*'} = (h_{01}^* \sum_{k=1}^{b_{1,j_1}} r_{1,j_1}^* + h_{02}^* \sum_{k=1}^{b_{2,j_2}} r_{2,j_2}^* + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} r_{i,j_i}^*) \\ \quad \cdot ar^{*'} bP. \end{cases}$$

$$\begin{cases} \alpha^* = (h_{01}^* \sum_{k=1}^{b_{1,j_1}} r_{1,j_1}^* + h_{02}^* \sum_{k=1}^{b_{2,j_2}} r_{2,j_2}^* + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} r_{i,j_i}^*), \\ \alpha'^* = (h_{01}^* \sum_{k=1}^{b_{1,j_1}} r_{1,j_1}'^* + h_{02}^* \sum_{k=1}^{b_{2,j_2}} r_{2,j_2}'^* + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} r_{i,j_i}'^*). \end{cases}$$

Computing (4) − (3), we obtain

$$abP = \frac{Y^* - Y'^*}{\alpha'^* - \alpha^*}.$$

Therefore, $\mathcal{B}$ can successfully solve the given instance of the CDH problem in $\mathbb{G}_1$.

3.5.3 Existential Unforgeability Against $\mathcal{A}_3$

Theorem 3.3 *If there exists an adversary $\mathcal{A}_3$ who can $(t, q_H{}^*, q_s, \epsilon)$-break the proposed scheme then there exists another algorithm $\mathcal{B}$ that can use $\mathcal{A}_3$ to solve an instance of the CDH problem in $\mathbb{G}_1$.*

Proof: Algorithm $\mathcal{B}$ is given a random instance (P, aP, bP) of the CDH problem in $\mathbb{G}_1$. Its goal is to compute abP by interacting with adversary $\mathcal{A}_3$ as described below. The hash function H_3 is regarded as the random oracles during the proof.
Setup:

- Run the **System Setup** algorithm to obtain the system parameters *params*.
- Set $pk_H = aP$, where aP is one of input of CDH problem.
- Maintain H_3-*list* that stores the results of queries to random oracles. Initially, it is empty.
- Return *params* and pk_H to $\mathcal{A}_3$.

H_3 **Queries**: Adversary $\mathcal{A}_3$ can make queries to the H_3 oracle of the inputs (m, ω, pol_C) at any time, $\mathcal{B}$ checks the H_3 list first:

- If there exists an item $((m, \omega, pol_C), h_3, r, coin)$ in the list, $\mathcal{B}$ will return h_3 to $\mathcal{A}_3$.
- Otherwise, $\mathcal{B}$ tosses a coin $coin \in \{0, 1\}$ such that $Pr[coin] = \delta$
  - If $coin = 1$, $\mathcal{B}$ chooses $r \in_R Z_P^*$ and computes $h_3 = rbP$.

– Otherwise, $coin = 0$, $\mathcal{B}$ chooses $r \in_R Z_P^*$ and computes $h_3 = rP$.

Signature Queries: In this process, $\mathcal{A}_3$ can ask at most q_S signature queries of his choice.

- If $coin$=0, then $\mathcal{B}$ chooses randomly $Y_{i,l} \in \mathbb{G}_1$, then computes

$$X_{i,l+1} = e(P, Y_{i,l}) \cdot \tau_{i,l}^{h_0} \cdot e\Big(P, D + arbP\Big),$$

$$Y_{i,j_i} = Y_i - h_0 \cdot \sum_{k=1}^{b_{i,j_i}} C_{i,j_i} - D - arbP.$$

- If $coin$=1, $\mathcal{B}$ terminates the simulation and reports failure.

Forgery: $\mathcal{A}_3$ will output a valid signature tuple $(m^*, \omega^*, X^*, \sigma^*)$ with successful probability at least ϵ.

- If $coin$=0, $\mathcal{B}$ aborts.

- If $coin$=1, this step is to apply the general forking lemma. $\mathcal{B}$ then replays $\mathcal{A}_3$ with the same tuple but different H_3. Suppose H_3 outputs h_3 and $h_3^{'}$ in the first round and second round, respectively. Here,

$$\begin{cases} \sigma^* = \Big([X_{i,1}^*, X_{i,2}^*, ..., X_{i,b_i}^*]_{1 \le i \le b}, Y^*\Big), \\ \sigma^{'*} = \Big([X_{i,1}^{'*}, X_{i,2}^{'*}, ..., X_{i,b_i}^{'*}]_{1 \le i \le b}, Y^{'*}\Big). \end{cases}$$

Where,

$$\begin{cases} Y^* = \Big((Y_1^* - h_{01}^* \cdot \sum_{k=1}^{b_i,j_1} C_{1,j_1}^* - D^* - a \cdot r^* bP + (Y_2^* - \\ \quad h_{02}^* \cdot \sum_{k=1}^{b_i,j_2} C_{2,j_2}^* - D^* - a \cdot r^* bP + ... + (Y_i^* - h_{0i}^* \cdot \\ \quad \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}^* - D^* - a \cdot r^* bP\Big) + \sum_{i=1}^{b} \sum_{j \ne j_i}^{b_i} Y_{i,l}^*, \\ Y^{*'} = \Big((Y_1^* - h_{01}^* \cdot \sum_{k=1}^{b_i,j_1} C_{1,j_1}^* - D^* - a \cdot r^{*'} bP) + (Y_2^* \\ \quad - h_{02}^* \cdot \sum_{k=1}^{b_i,j_2} C_{2,j_2}^* - D^* - a \cdot r*' bP) + ... + (Y_i^* - \\ \quad h_{0i}^* \cdot \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}^* - D^* - a \cdot r^{*'} bP)\Big) + \sum_{i=1}^{b} \sum_{j \ne j_i}^{b_i} Y_{i,l}^*. \end{cases} \tag{3.3}$$

$$\begin{cases} Y^* = (Y_1^* + Y_2^* + ... + Y_i^*) - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} C_{1,j_1} + h_{02}^* \sum_{k=1}^{b_{2,j_2}} \\ \quad C_{2,j_2} + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}) - i \cdot D^* - i \cdot ar^* bP \\ \quad + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*, \qquad (5) \\ Y'^* = (Y_1^* + Y_2^* + ... + Y_i^*) - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} C_{1,j_1} + h_{02}^* \sum_{k=1}^{b_{2,j_2}} \\ \quad C_{2,j_2} + ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}) - i \cdot D^* - i \cdot ar'^* bP \\ \quad + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*. \qquad (6) \end{cases}$$

$$\begin{cases} i \cdot ar^* bP = \sum_{i=1}^{b} Y_i^* - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} C_{1,j_1} + h_{02}^* \sum_{k=1}^{b_{2,j_2}} C_{2,j_2} + \\ \quad ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}) - Y^* - i \cdot D^* + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*, \\ i \cdot ar^{*'} bP = \sum_{i=1}^{b} Y_i' - (h_{01}^* \sum_{k=1}^{b_{1,j_1}} C_{1,j_1} + h_{02}^* \sum_{k=1}^{b_{2,j_2}} C_{2,j_2} + \\ \quad ... + h_{0i}^* \sum_{k=1}^{b_{i,j_i}} C_{i,j_i}) - Y^{*'} - i \cdot D^* + \sum_{i=1}^{b} \sum_{j \neq j_i}^{b_i} Y_{i,l}^*. \end{cases}$$

Computing (5) − (6), we obtain

$$abP = \frac{Y^* - Y'^*}{i(r'^* - r^*)}.$$

Therefore, $\mathcal{B}$ can successfully solve the given instance of the CDH problem in $\mathbb{G}_1$.

3.5.4 Credentials Ambiguity

Our policy-based signature scheme achieves credentials ambiguity in the random oracle model. The proof is similar to the proof given in [LW04] [ZK02]: for all indices i, Y_i is chosen randomly in $\mathbb{G}_1$, so x_{i,j_i} is uniformly distributed in G_2. Similarly, for all indices i, l, $Y_{i,l}$ is chosen randomly in $\mathbb{G}_1$, it leads to that all $x_{i,l}$

are uniformly distributed in $\mathbb{G}_2$. Thus, given a message m and the signature $\sigma = \Big([X_{i,1}, X_{i,2}, ..., X_{i,b_i}]_{1 \leq i \leq b}, Y\Big)$ on m according to pol_C, σ does not reveal which credentials have been used to generate it.

3.6 Summary

In this section, a policy-based authentication scheme was proposed to provide a solution to mobile agent security. In our scheme, only the host, who meets all the requirements in the set of policies defined by an agent owner, can be included in the agent network and generate a valid policy-based signature. That is, hosts are authenticated by an agent owner via a set of defined policies, and then the agent owner authorizes the host with a warrant. The proposed scheme is useful for practical applications. Based on our security model, we proved that our scheme is secure against the strongest adversaries.

Chapter 4

Policy-based Encryption with Much Smaller Ciphertext and Application in Multi-agent Systems

4.1 Introduction

Software agents are entities that have special properties, such as autonomy and certain intelligence. A multi-agent system consists of multiple autonomous agents with some characteristics: (1) each agent can not solve a problem by himself; (2) data is decentralized; (3) the computation is asynchronous. Consequently, in order to ensure frequent secure communications within a multi-agent system, the following security properties should be provided:

- Confidentiality: assurance that unauthorized parties are not able to accessible to the communicated information.
- Authentication: assurance that a communication originates from its claimant.
- Data integrity: assurance that unauthorized parties can not tamper with the communicated information.

Unfortunately, most existing research only pay attention to the security protection of the message exchanged among the agents. There was little work focusing on authentication and privacy issues of multi-agent systems. In open environment, if an agent cannot trust another agents, its privacy must be protected.

Consider the following scenario: in a multi-agent system, all agents in the group are distinguished by different properties. The relationship among the agents within the group can be partners or competitors. In the real world, this could be a situation in electronic commerce, where agents are classified into buyers and sellers. Each seller can be distinguished by different goods, price, credit rating, *et al*. Each buyer

can be distinguished by different goods offers, price, *et al*. When two agents need to exchange the information securely with each other, neither one is willing to reveal his identity firstly. Or neither one wants to send a message directly to any agent without identity authentication. The seller does not want to disclose the offer to other sellers, and in order to protect privacy, the buyer also does not want to disclose his offer to other sellers who cannot supply the target goods. We allow both sellers and buyers, namely, all agents involved in the system to hold different policies according to their different characteristics. The seller's offer can be only sighted when the policies held by buyers match the policies embedded in the offer data set. In this algorithm, both buyers and sellers know that they are communicating with a legitimate entity.

At the first glance, we may consider there exist several solutions which are suitable for the scenario. In 2003, Balfanz *et al*. [BDS+03] proposed a scheme called secret handshakes. Two communicating entities can mutually authenticate by using their credentials from the same credentials issuer. Later, Holt, Bradshaw and Seamons [HBSO03] proposed a concept called hidden credentials, which allows the sender to encrypt the message under the receiver's credentials only, the sender does not need have any credential for himself. However, this paper does not present a concrete scheme to show how it works. Li, Du and Boneh [LDB03] proposed a scheme called oblivious signature-based envelop (OSBE). OSBE enables a sender to encrypt a message, send the ciphertext to a receiver who can decrypt it if and only if the receiver has the valid signature from TA. But with a closer look at OSBE, it requires senders and receivers to have an agreement on the credentials which are used for decrypting sender's ciphertext by the receiver. It can not be avoided that senders encrypt a message and return it to receivers without disclosing credentials he must use to decrypt the ciphertext. This can be very significant if the credentials in question are extremely sensitive.

Policy-based encryption schemes can address the problem of multi-agents authentication and privacy protection. Bagga and Molva [BM05, BM06] firstly presented the concept of policy-based cryptography and proposed a concrete policy-based encryption scheme. A policy specifies the constrains under which a specific action can be performed on a certain sensitive resource. Policy-based encryption (PBE) allows a sender to encrypt a message with respect to a credential-based policy so that only the entities who are compliant with the policy can successfully decrypt the ciphertext. However, the scheme in [BM06] is not suitable for our system. End receivers are designated by their ID information. The sender uses receivers' public

keys during the encryption process. Only the designated receiver who holds the credentials matching the policy and also corresponding private key can access to the decryption process. Furthermore, the policy expression in [BM06] is in conjunctive-disjunctive normal form, like $pol = \wedge_{i=1}^{a}[\vee_{j=1}^{a_i}[\wedge_{k=1}^{a_{i,j}} < A_{i,j,k} >]]$, where pol denotes a set of policies, if we set $a_{i,j} = 1$ that transforms the formula into to conjunctive normal form $pol = \wedge_{i=1}^{a}[\vee_{j=1}^{a_i} < A_{i,j} >]$. In this case, the size of encryption key pol and decryption key credentials set $Cred_{j_i,...,j_i}(pol)$ are both $(a \times a_i)$, and the size of ciphertext $C = \{U, [V_{i,1}, V_{i,2}, ..., V_{i,a_i}], W\}$ is still $(a \times a_i + 2)$ which will increase linearly in pol in [BM06]. Obviously, it is a significant issue in less communication channel.

Motivation and Contribution: We focus on a scheme with smaller ciphertext size and lower computational cost. In this section, we propose a new policy-based encryption scheme with shorter key size and constant ciphertext size and apply it into the multi-agent systems, which allows policy-based authenticated communications and provides privacy protection to agents. The key encapsulation mechanism (KEM) is constructed in our scheme so that long messages can be encrypted with a short session key. In our solution, the size of ciphertext is fixed even if the number of policies is increasing. In [BM06], the size of ciphertext is $(a \times a_i + 2)$, however, the ciphertext size in our system is $(2a + 1)$. When $a_i \geq 2$, especially the policy set is growing larger, our scheme is more efficient than Bagga' scheme. Our credentials system is based on the short signature scheme described in [BLS04] and policy-based encryption scheme extends the ID-based broadcast encryption defined in [Del07]. We also manipulate the policies that are formalized as monotonic logic expressions involving complex disfunctions of conditions which are more realistic for agents. The credentials have to be kept secretly by their owners. Any receiver if he/she has the corresponding credentials can decrypt the ciphertext. We also define a rigorous security model for multi-agent transaction, which captures the most powerful attacks including adaptive chosen message attacks. The security of our scheme is based on the hardness of $((p, g, F)$-GDDHE) problem in the random oracle model.

4.2 Definition

4.2.1 Policy, Assertion and Credential

Policy *pol* is fulfilled by a set of credentials generated by one trusted authority. In order to ensure a host to be authorized to perform a special action on a sensitive resource, he has to prove his compliance to the policy defined by the agent owner. He has to prove that he possesses a minimal set of credentials that is required by *pol* to permit action on the resource. Let TA denote a trusted authority who issues credentials associated with policies. Let A denote an assertion which may be a hash value of some statement, such as "Virginblue agency". In general, a policy *pol* can be represented in the disjunctive normal form (DNF) or the conjunctive normal form (CNF). Whenever TA is asked to sign an assertion $A \in \{0,1\}^*$, it first checks the validity of A. If A is valid, then TA executes algorithm **IssueCredential** defined below and returns the credential to the requester.

4.2.2 A High Level Description of Our Scheme

We assume that agents reside in different hosts in network environment are classified into sellers and buyers. They are managed by a trusted authority (TA) who issues the credentials corresponding to a set of policies to credential questers. When an agent sends a message to other agents, it encrypts the message with the selected policies, only those who hold the corresponding credentials can read the encrypted message.

Figure 4.1 demonstrates how our protocol works, where five agents, labeled from 1 to 5, reside in five corresponding hosts. All agents represent a buyer or a seller in a real electronic commerce system. Assume that agent 4 wants to buy a flight ticket for a trip to the USA on the 10th, March, 2011 and the price needs to be below 85 dollars. The flight ticket information can be considered as a selected policy. Agent 4 sends a message which is encrypted by its selected policies. Only the agents who possess the credentials matching the policies are able to decrypt the encrypted message and then authenticate each other. In this example, we assume agent 1 holds the correct credentials for selling flight ticket to the USA on the 10th, March, 2011 and it costs 80 dollars.

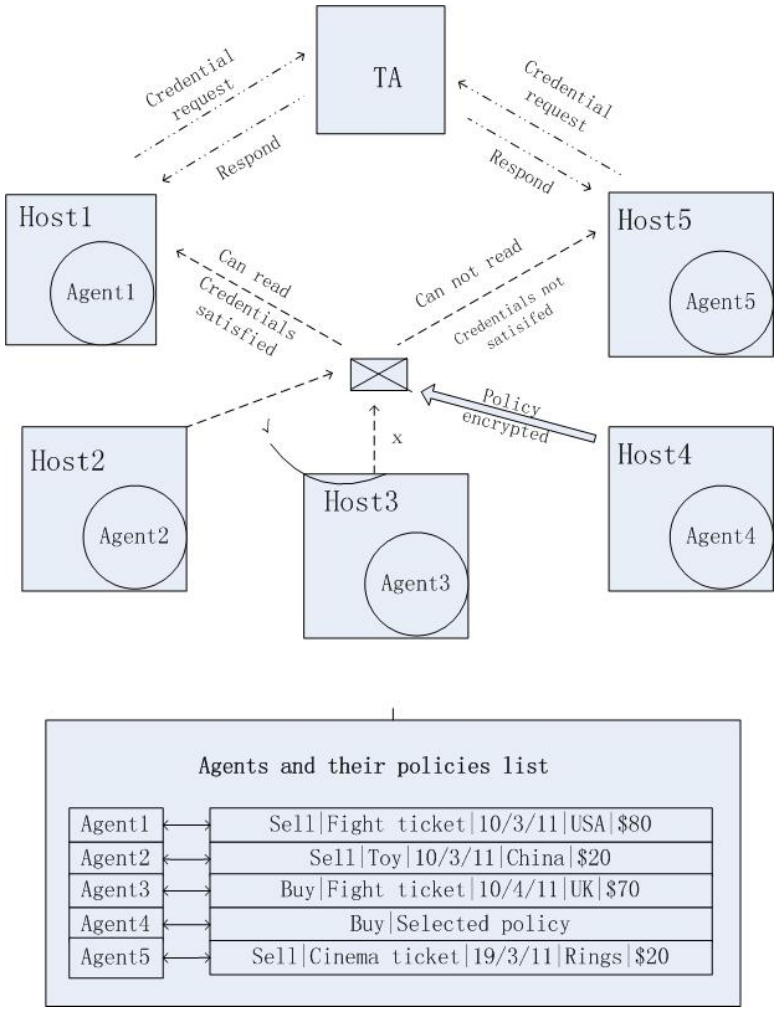

Figure 4.1: Multi-agent model

4.2.3 Definition of PBE Scheme

Our scheme includes following algorithms: **Setup**, **IssueCredential**, **SendMsg** and **DecrptMsg**. We define the policy in a conjunctive normal form (CNF).

- **Setup**: This algorithm takes a security parameter k as input, and outputs the system parameters $Params$ and TA master key (γ, g). That is,

$$Setup(1^k) \rightarrow (Params, \gamma, g)$$

- **IssueCredential**: TA runs the algorithm by taking $Params$, a policy pol_i, (γ, g) as input and outputs a credential $Cred_i$ which is respect to the policy

pol_i. That is,

$$IssueCredential(Params, (\gamma, g), pol_i) \rightarrow (Cred_i)$$

- **SendMsg**: The sender runs this algorithm by taking $Params$, a selected policy pol_i (in conjunctive normal form), a message m, γ as input, and outputs an ciphertext C. That is,

$$SendMsg(Params, pol, m, \gamma) \rightarrow (C)$$

- **DecrptMsg**: The receiver runs this algorithm by taking $Params$, the ciphertext C, the policy pol, the corresponding policy credentials $Cred1, ..., Cred_l$, γ as input, and outputs a message m where m is the plaintext of C, if and only if $Cred1, ..., Cred_l$ match the policy set embedded in C. That is,

$$DecrptMsg(Params, pol, Cred1, ..., Cred_l, \gamma, C) \rightarrow (m)$$

4.2.4 Security Model

Our scheme is an extension from IBBE [Del07], which means it is required that an policy-based encryption scheme also satisfies the strong notion of security defined in [Del07]. We define an adaptive chosen ciphertext attacker in our scheme as adversary $\mathcal{A}$. Chosen ciphertext attacks for PBE are defined in terms of an interactive game, played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as following:

- Setup: Take a security parameter k as input, the challenger runs the algorithm **Setup** to obtain the system parameters $Params$ and the master key (γ, g). Finally, the challenger $\mathcal{C}$ gives the adversary $\mathcal{A}$ $Params$ and keeps (γ, g) as secret.

- Phase 1:

  - IssueCredential queries: For a credential $Cred_i$ of policy pol query, $\mathcal{C}$ runs the algorithm **IssueCredential** to obtain $Cred_i$ of pol and returns it to $\mathcal{A}$.

- Challenge: When $\mathcal{A}$ decides that Phase 1 is over, he outputs two equal length messages M_0, M_1, and a policy POL_{ch} he wishes to be challenged. It is assured that at least one of the policies has not been submitted in **IssueCredential**

queries in Phase 1. $\mathcal{C}$ then selects a random bit $b \in 0, 1$ and runs the algorithm **SendMsg** by taking POL_{ch}, M_b as input and obtains the ciphertext C. The challenger $\mathcal{C}$ sends C to the adversary $\mathcal{A}$ then.

- Phase 2: Upon receiving the challenging ciphertext C, $\mathcal{A}$ can still make **IssueCredential** queries adaptively as in Phase 1 except that at least one of the policy in *pol* must not be submitted in **IssueCredential** queries.
- Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in 0, 1$ and wins the game if $b = b'$.

4.2.5 Pairing and Complexity Assumption

Bilinear Map: Let G_1, G_2 and G_T be three cyclic groups of prime order p. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing with the following properties:

- Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- Non-degenerate: There exists $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2) \neq 1$;
- Computability: There is an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}_1$.

The General Diffie-Hellman Exponent Assumption

The generalization of the Diffie-Hellman exponent assumption was proposed by Boneh, Boyen and Goh [BBG05]. They introduced a class of assumptions which includes a lot of assumptions that appeared with new pairing-based scheme. It includes DDH, BDH, q-BDHI and q-BDHE assumptions.

Definition 4.1 *((f, g, F)-GDDHE). let $\mathcal{B} = (p, G_1, G_2, G_t, e(., .))$ be a bilinear map group system and let f and g be two coprime polynomials with pairwise distinct roots, of respective orders t and n. Let g_0 be a generator of G_1 and h_0 a generator of G_2. Solving the (f, g, F)-GDDHE problem consists, given*

$$g_0, g_0^{\gamma}, ..., g_0^{\gamma^{t-1}}, \qquad g_0^{\gamma \cdot f(\gamma)}, \qquad g_0^{k \cdot \gamma \cdot f(\gamma)}$$

$$h_0, h_0^{\gamma}, ..., h_0^{\gamma^{2n}}, \qquad h_0^{k \cdot g(\gamma)},$$

and $T \in G_T$, in deciding whether T is equal to $e(g_0, h_0)^{k \cdot f(\gamma)}$ or to some random element of G_T [Del07].

4.3 Our Scheme

We define the policy $pol = \bigwedge_{i=1}^{a}(\bigvee_{j=1}^{a_i} A_{i,j})$ and assume that an agent $agent_1$ uses the policy pol to encrypt a message. For any agent $agent_i$ who holds the credentials set satisfying the corresponding policy pol can decrypt the ciphertext. The scheme is described as follows:

Setup:

- Take a security parameter k as input, the algorithm outputs the system public information $(p, G_1, G_2, e(.,.))$, Let g be the generator of G_1, and h be the generator of G_2.
- Choosing a hash function $H : \{0,1\}^* \rightarrow Z_p^*$ and randomly choose $\gamma \in Z_q^*$.
- Set the Third Party (TA) master key $s = (g, \gamma)$.
- Set the public key $pk = (g^\gamma, e(g,h), h, h^\gamma, ..., h^{\gamma^m})$.

IssueCredential: Each agent $agent_i$ can request the credentials corresponding to his policy from TA. TA returns the credential which is signed by his the master key (g, γ).

$$Cred_{A_{i,j}} = g^{\frac{1}{\gamma + H(A_{i,j})}}$$

Encrypt: Given $PK = (g^\gamma, e(g,h), h, h^\gamma, ..., h^{\gamma^m})$, the policy set $pol = \bigwedge_{i=1}^{a}(\bigvee_{j=1}^{a_i} A_{i,j})$, the agent $agent_1$ wants to encrypt the message with a selected policy pol as follows:

- For i=1,...,a, where a is the index of policy pol. (That is, each disjunctive block in pol);
  - Pick a random number $k_i \in z_q^*$,
  - Compute

$$K_i = e(g,h)^{k_i}$$

$$C_i = g^{\gamma(-k_i)}$$

$$D_i = h^{k_i \prod_{j=1}^{a_i}(\gamma + H(A_{i,j}))}$$

- Compute $K = \prod_{i=1}^{a} K_i$.

- Encrypt the message m with using the session key K, the ciphertext

$$\mathcal{C} = (C_i, D_i, m \bigoplus K).$$

- Broadcast C.

Decrypt: Suppose an agent who holds the credentials set $\{Cred_{A_{1,j_1}}, Cred_{A_{2,j_2}}, ..., Cred_{A_{a,ja}}\}$ which satisfy $agent_1$'s policy pol. Before he decrypting the message m, he must retrieve the message encryption session key K. The user does as follows:

- For i=1,...,a, where a is the index of policy pol. (That is, each disjunctive block in pol);
  - compute

$$\rho(\gamma)_i = \frac{1}{\gamma}\Big(\prod_{j=1, j\neq j_i}^{a_i}(\gamma + H(A_{i,j})) - \prod_{j=1, j\neq j_i}^{a_i} H(A_{i,j})\Big)$$

  - compute

$$K_i = (e(C_i, h^{\rho(\gamma)_i}) \cdot e(Cred_{A_{i,j_i}}, D_i))^{\frac{1}{\prod_{j=1, j\neq j_i}^{a_i} H(A_{i,j})}}$$

- Compute $K = \prod_{i=1}^{a} K_i$.
- Compute $m = C \bigoplus K$.

Correctness:

$$\begin{aligned} K_i' &= e(C_i, h^{\rho(\gamma)_i}) \cdot e(Cred_{A_{i,j_i}}, D_i) \\ &= e(g^{-k_i\gamma}, h^{\rho(\gamma)_i}) \cdot e(g^{\frac{1}{\gamma + H(A_{i,j_i})}}, h^{k_i \prod_{j=1}^{a_i}(\gamma + H(A_{i,j}))}) \\ &= e(g,h)^{-k_i(\prod_{j=1, j\neq j_i}^{a_i}(\gamma + H(A_{i,j})) - \prod_{j=1, j\neq j_i}^{a_i} H(A_{i,j}))} \cdot e(g,h)^{k \prod_{j=1, j\neq j_i}^{a_i}(\gamma + H(A_{i,j}))} \\ &= e(g,h)^{k \prod_{j=1, j\neq j_i}^{a_i} H(A_{i,j})} \end{aligned} \tag{4.1}$$

Thus $K_i'^{\frac{1}{\prod_{j=1, j\neq j_i}^{a_i} H(A_{i,j})}} = K_i$.

4.4 Efficiency

In this section, we compare the ciphertext size and the computational cost between the proposed scheme and the policy-based encryption scheme proposed by Bagga

and Molva [BM06]. The computational cost of both schemes includes the encryption cost and decryption cost.

In [BM06], when we take $a_{i,j} = 1$, the $pol = \bigwedge_{i=1}^{a}[\bigvee_{j=1}^{a_i}[\bigwedge_{k=1}^{a_{i,j}}]] = \bigwedge_{i=1}^{a}(\bigvee_{j=1}^{a_i} A_{i,j})$, which is the same as the policy expression in our scheme. The message m is encrypted to $(U, [v_{i,1}, v_{i,2}, ..., v_{i,a_i}]_{1\leq i\leq a}, \omega)$, which there has been $(a \cdot a_i + 2)$ elements. Here, a and a_i are the indexes, which represent the number of policy attributes. In our scheme, the ciphertext is (C_i, D_i, K), which there has been $(2a + 1)$ elements. Obliviously, when $a_i \geq 2$, our scheme is much more efficient.

Table 4.1: Bandwidth comparison between two schemes

Scheme	Ciphertext Length
Scheme in [BM06]	$a \cdot a_i + 2$
Our scheme	$2a + 1$

Table 4.2: Computational costs in terms of pairing computation comparison between two schemes

Scheme	PolEnc	PolDec
Scheme in[BM06]	$a \cdot a_i$	a
Our scheme	a	$2a$

In the Bagga's scheme, there are $a \cdot a_i$ pairings in encryption phase and a pairings in decryption phase. However, in our scheme, there are a pairings in encryption phase and $2a$ pairings in decryption phase.

4.5 Security Analysis

The PBE scheme is IND-Pol-CPA secure in the random oracle model under the assumption that (f, g, F)-GDDHE problem is hard.

Theorem 4.1 *If there exists an adversary $\mathcal{A}$ who can (t, q_H, ϵ)-break the proposed scheme then there exists another algorithm $\mathcal{B}$ who can use $\mathcal{A}$ to solve an instance of the (f, g, F)-GDDHE problem.*

Proof:

Both the adversary and the challenger are given as input n, and t the total number of IssueCredtial queries and random oracle queries that can be issued by the adversary.

Algorithm $\mathcal{B}$ is given as input system $\mathcal{I} = (p, G_1, G_2, G_T, e(.,.))$, and a (f, g, F)-GDDHE instance in $\mathcal{I}$. We thus have f and g two coprime polynomials with pairwise distinct roots, of respective orders t and n, and $\mathcal{B}$ is given

$$g_0, g_0^{\gamma}, ..., g_0^{\gamma^{t-1}}, \qquad g_0^{\gamma \cdot f(\gamma)}, \qquad g_0^{k \cdot \gamma \cdot f(\gamma)}$$

$$h_0, h_0^{\gamma}, ..., h_0^{\gamma^{2n}}, \qquad h_0^{k \cdot g(\gamma)},$$

and $T \in G_T$, in deciding whether T is equal to $e(g_0, h_0)^{k \cdot f(\gamma)}$ or to some random element of G_T.

For simplicity, we state that f and g are unitary polynomials, but this is not a mandatory requirement.

Notations

- $f(X) = \prod_{i=1}^{t}(X + x_i), g(X) = \prod_{i=t+1}^{t+n}(X + x_i)$
- $f_i(x) = \frac{f(x)}{x+x_i}$ for $i \in \{1, t\}$, which is a polynomial of degree $t - 1$
- $g_i(x) = \frac{g(x)}{x+x_i}$ for $i \in \{t + 1, t + n\}$, which is a polynomial of degree $n - 1$

Setup: Run the **Setup** algorithm to obtain the system parameters $Param$, $\mathcal{B}$ sets

$$g = g_0^{f(\gamma)}$$

$$g^{\gamma} = g_0^{\gamma \cdot f(\gamma)}$$

$$h = h_0^{\prod_{i=t+s^*+1}^{t+n}(\gamma + x_i)}$$

$$e(g, h) = e(g_0, h_0)^{f(\gamma) \cdot \prod_{i=t+s^*+1}^{t+n}(\gamma + x_i)}$$

$\mathcal{B}$ then defines the public key as

$$PK = (g_0^{\gamma \cdot f(\gamma)}, e(g_0, h_0)^{f(\gamma) \cdot \prod_{i=t+s^*+1}^{t+n}(\gamma + x_i)}, h, h^{\gamma}, ..., h^{\gamma^n}).$$

Note $\mathcal{B}$ can by no means compute the value of g. $\mathcal{B}$ runs $\mathcal{A}$ on the system parameters $Param$ and PK, H is a random oracle.

Hash Queries: Proceeding adaptively, the adversary $\mathcal{A}$ can make queries to the H oracle of input $(A_{i,j})$ at any time, $\mathcal{B}$ checks the H-list of tuple $(A_{i,j}, x_i)$ as follows:

- If there exists an item $(A_{i,j}, x_i)$ in the list, $\mathcal{B}$ will return x_i to $\mathcal{A}$.

- Otherwise, $\mathcal{B}$ sets $H(A_{i,j}) = x_i$, and adds $(A_{i,j}, x_i)$ to the H-list.

Phase 1: For each query $A_{i,j}/Cred_{i,j}$ issued by adversary $\mathcal{A}$. $\mathcal{B}$ responds to this query as follows:

- The challenger runs **IssueCrendial** on $A_{i,j}$ and forwards the resulting credential to the adversary. To generate the Credential,
  - If $\mathcal{A}$ has already issued an credential query on $A_{i,j}, \mathcal{B}$ responds with the corresponding $Cred_{A_{i,j}}$ in the list $\mathcal{L}_H$.
  - Else, if $\mathcal{A}$ has already issued a hash query on $A_{i,j}$, then $\mathcal{B}$ uses the corresponding x_i to compute $Cred_{A_{i,j}} = g_0^{f_i(\gamma)} = \frac{1}{\gamma + H(A_{i,j})}$. One can verify that $Cred_{A_{i,j}}$ is a valid credential. $\mathcal{B}$ then completes the list $\mathcal{L}_H$ with $Cred_{A_{i,j}}$ for $A_{i,j}$.
- Otherwise, $\mathcal{B}$ sets $H(A_{i,j}) = x_i$, computes the corresponding $Cred_{A_{i,j}}$ exactly as above, and adds it into the list $\mathcal{L}_H$ for $A_{i,j}$.

Challenge: When $\mathcal{A}$ decides that Phase 1 is over, algorithm $\mathcal{B}$ runs algorithm **Encrypt** to obtain (C_1, D_1, K)

$$C_1 = g_0^{-k\cdot\gamma\cdot f(\gamma)} \quad , C_2 = h_0^{k\cdot g(\gamma)}, \quad K = T^{\prod_{i=t+s^*+1}^{t+n} x_i} \cdot e(g_0^{k\cdot\gamma\cdot f(\gamma)}, h_0^{(}q(\gamma)))$$

with $q(\gamma) = \frac{1}{\gamma}(\prod_{i=t+s^*+1}^{t+n}(\gamma + x_i) - \prod_{i=t+s^*+1}^{t+n} x_i)$. One can verify that:

$$C_1 = \omega^{-k}, \qquad C_2 = h_0^{k\cdot\prod_{i=t+s^*+1}^{t+n}(\gamma+x_i)\cdot\prod_{i=t+1}^{t+s^*}(\gamma+x_i)} = h^{k\cdot\prod_{i=t+1}^{t=s^*}(\gamma+H(ID_i^*))}.$$

Note if $T = e(g_0, h_0)^{k\cdot f(\gamma)}$, then $K = e(g, h)^k$.

The challenger then randomly selects $b \in \{0, 1\}$, sets $K_b = K$, and sets K_{1-b} to a random value in $\mathcal{K}$. The challenger returns (C_1, D_1, K_0, K_1) to $\mathcal{A}$.

Phase 2: The adversary continues to issue queries $q_{C_{m+1}}, ..., q_{C_E}$ where q_i is an credential query $A_{i,j}$ with the constraint that $A_{i,j}$ has not been queried before.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

4.6 Summary

We presented a secure communication scheme for multi-agent systems in untrusted network. Compared with other encryption schemes [BM05, BM06] proposed before,

our scheme captures both data confidentiality and user privacy. Furthermore, our ciphertext size is much smaller and more efficient than current PBE schemes. Policy-based authentication was implemented in our scheme in order to provide secure interactions among agents and hosts. We defined our scheme which is IND-Pol-CPA secure in random oracle based on the hardness of $((p, g, F)$-GDDHE) problem and provided rigorous security proof.

Chapter 5

Conclusion

In this thesis, we fucus on the security issues about software agents. Non-repudiation of the contract, delegation abuse in mobile agent systems, and agents privacy protection in multi-agent systems during E-commerce applications are captured. We proposed two concrete policy-based schemes to provide security mechanism for agents. Our main contributions in the thesis can be summarized as follows:

In Chapter 3, we proposed a novel policy-based authentication scheme for mobile agents. Our scheme eliminates the non-repudiation problem in the way that visiting host of an agent acts as a proxy signer who can sign an offer. The host can not deny the signature with respect to the delegation abuse issue. The scheme allows the owner of mobile agent to select a set of remote hosts that will execute the mobile agent according to a set of policies. Only those hosts that satisfy the selected policies can execute the mobile agent and sign the contract later. The selected hosts list can be updated dynamically by the agent owner without any additional computational cost. We also provided the security model and rigorous security proofs related to scheme. The security of our scheme is based on the hardness of Computational Diffie-Hellman problem in the random oracle model.

In Chapter 4, a new policy-based encryption scheme was presented to provide a secure communicating channel for multiple agents in open untrusted environment. It allows an agent to encrypt the information with a selected policy set so that only the agent who holds the credentials that satisfy the policy set can perform the decryption. Our scheme provides not only the data confidentiality but also privacy. Compared with the first PBE scheme proposed by Bagga and Molva [BM06], the size of ciphertext in our scheme is much smaller and the computation is much more efficient. We defined an elaborated security model for multi-agent transaction, which captures the most powerful attacks including adaptive chosen message attacks. The security of our scheme is based on the hardness of $((p, g, F)$-GDDHE) problem in

the random oracle model.

Both of the proposed schemes in the thesis enjoy the security requirements of agents, which are data confidentiality and integrity, authentication of origin and availability of agents. Future work will focus on how to improve the performance of our proposed schemes. In particular we will explore ways of reducing computational cost and extend the properties satisfied by the schemes.

Bibliography

[ACCK01] Joy Algesheimer, Christian Cachin, Jan Camenisch, and Gunter Karjoth. Cryptographic security for mobile code. *2001 IEEE Symposium on Security and Privacy,*, pages 2–11, 2001.

[AL04] Amit K Awasthi and Sunder Lal. A new proxy ring signature scheme. *Arxiv preprint cs/0410010*, 2004.

[AL05] Amit K Awasthi and Sunder Lal. Id-based ring signture and proxy ring signature schemes from blinear pairings. *Arxiv preprint cs/0504097*, 2005.

[BB04] Don Boneh and Xavier Boyen. Efficient selective-id secure identity based encryption without random oracles. *In Advances in Cryptology, Proc. EUROCRYPT2004, LNCS*, 3207:223–238, 2004.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. *In Cramer, R.J.F. (Es.) EUROCRYPT 2005, LNCS*, 3494:440–456, 2005.

[BC02] Elmarie Bierman and Elsabe Cloete. Classification of malicious host threats in mobile agent computing. *In Proc. of SAICSIT2002*, pages 141–148, 2002.

[BDS+03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Hessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. *In IEEE Symposium on Security and Privacy*, pages 180–196, 2003.

[BDS+04] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based

key agreements. *P.J. Lee (Ed.): ASIACRYPT 2004, LNCS, Springer Verlag*, 3329:293–307, 2004.

[BF01] Don Boneh and M. Franklin. Identity-based encryption from the weil pairing. *In J. Kilian, Editor, Adcances in Cryptology, Proc. CRYPTO 2001, LNCS, Springer Verlag*, 2139:213–229, 2001.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. *In Shoup, V., Editor, CRYPTO 2005, LNCS, Springer Verlag*, 3621:258–275, 2005.

[BHS04] Robert W. Bradshaw, Jason E. Holt, and Kent E. Seamons. Concealing complex policies with hidden credentials. *In Proceedings of 11th ACM Conference on Computer and Communications Security*, pages 146–157, 2004.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[BM05] Waild Bagga and Refik Molva. Policy-based cryptography and applications. *Financial Cryptography and Data Security, LNCS*, 3570:72–87, 2005.

[BM06] Waild Bagga and Refik Molva. Collusion-free policy-based encryption. *ISC, LNCS*, 4176:233–245, 2006.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *In Proc. of First Annual Conference on Computer and Communications Security, ACM*, 1993.

[CC03] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. *In PKC2003, LNCS, Springer Verlag*, 2003.

[CEdG88] David Chaum, Jan-Hendrik Evertse, and Jerven Van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generlizations. *In Advances in Cryptology, Proc. EUROCRYPT87, LNCS, Springer Verlag*, 304:128–141, 1988.

[CH91] David Chaum and Van Heylst. Group signature. *In Proceeding of EUROCRYPT'91 of the 10th Annual International Conference on Theory*

and Application of Cryptographic Techniques, Springer Verlag, 1991, 1991.

[Che98] David M. Chess. Security issues in mobile code systems. *Mobile Agents and Security*, 1949:1–14, 1998.

[CJT04] Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik. Secret handshakes from ca-oblivious encryption. *In ASIACRYPT2004, Springer*, 3329:293–307, 2004.

[COS86] Don Coppersmith, Andrew Odlyzko, and Richard Schroeppel. Discrete logarithms in gf(p). *Algorithms*, 1(1):1–15, 1986.

[CP95] L. Chen and T.P. Pedersen. New group signature schemes. *Advances in Cryptology-EUROCRYPT'94, LNCS*, 950:171–181, 1995.

[CPV03] Joris Claessens, Bart Preneel, and Joos Vandewalle. How can mobile agents do secure electronic transaction on untrust hosts? a survey of the security issues and the current solutions. *ACM Transactions on Internet Technology*, 3(1):28–48, February 2003.

[CS03] Jan Camenisch and Victor Shoup. Practical verificable encryption and decryption of discrete logarithms. *In Advances in Cryptology, Proc. CRYPTO2003, LNCS, Springer Verlag*, 2729:126–144, 2003.

[Del07] Cècile Delerablèe. Identity-based broadcast encryption with constand size ciphertext and private keys. *In ASIACRYPT'07 Proceedings of the Advances in Crypotology 13th international conference on Theory and application of cryptology and information security , Springer-Verlag*, 2007.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, 22:644–654, 1976.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. *In Stinson, D.R., Editor, CRYPTO 1993, LNCS, Springer Verlag*, 773:480–491, 1994.

[GK94] Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Commun. ACM*, 1994.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 1988.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. *In Proceedings of the 13th ACM conference on Computer and Communications Security*, 2006.

[Gri01] Martin L. Griss. Software agents as next generation software components. *Component-based software engineering*, 2001.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertext). *A. Joux (Ed.): EUROCRYPT 2009, LNCS*, 5479:171–188, 2009.

[HBSO03] Jason E. Holt, Robert W. Bradshaw, Kent E. Seamons, and Hilarie Orman. Hidden credentials. *In Proceedings of the 2003 ACM workshop on Privacy in the Electronic Society*, 2003.

[HLR10] Javier Herranz, Fabien Laguillaumie, and Carla Rafols. Constant size ciphertexts in threshold attribute-based encryption. *PKC2010, LNCS, International Association for Cryptology Research*, 6056:19–34, 2010.

[Hoh98] Fritz Hohl. Time limited blackbox security: Protecting mobile agent from malicious hosts. *Mobile Agents and Security*, 1419:98–113, 1998.

[Hoh8b] Fritz Hohl. Time limited blackbox security: protecting mobile agents from malicious hosts. *In Mobile Agents and Security, G. Vigna, Ed., Lecture Notes in Computer Science, Springer-Verlag*, 1419:92–113, 1998b.

[HS02] Dani Halevy and Adi Shamir. The lsd broadcast encryption scheme. *In M. Yun, Editor, CRYPTO 2002, LNCS, Springer Verlag*, 2442:47–60, 2002.

[Jan99] W.A. Jansen. Mobile agents and security. *NIST Special Publication*, pages 800–19, 1999.

[Jan00] W.A. Jansen. Countermeasures for mobile agent security. *Comput. Commun*, 23(17):1667–1676, Nov. 2000.

[JN] A. Joux and K. Nguyen. Separte decision diffie-hellman from deffie-hellman in cryptographic groups. *ePrint.iacr.org*.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. *In Wieb Bosma, editor, Proc. of the ANTS-IV conference, LNCS, Springer Verlag*, pages 385–394, 2000.

[Jou02] Antoine Joux. The weil and tate pairing as building blocks for public key cryptosystems survey. *Algorithmic nubmer theory, LNCS*, 2362:11–18, 2002.

[KBC00] Panayiotis Kotzanikolaou, Mike Burmester, and Vassilios Chrissikopoulos. Secure transactions with mobile agents in hostile environments. *Information Security and Privacy-ACISP*, 1841:289–297, 2000.

[KC02] Seongyeol Kim and Ilyong Chung. A secure mobile agent system applying identity-based digital signature scheme. *EurAsia-ICT, LNCS*, 2510:588–596, 2002.

[KCK99] Panayiolaou Kotzanikolaou, Vassilios Chrissikopoulos, and George Katsirelos. Mobile agents for secure electronic transactions. *In Recent Advances in Singal Processing and Communicatons, N. Mastorakis, Ed., World Scientific, River Edge, NJ*, pages 363–368, 1999.

[KJLK01] H. Kim, J.Baek, B. Lee, and K. Kim. Secret computation with secrets for mobile agent using on-time proxy signature. *the 2001 Symposium on Cryptography and information security (SCIS)*, January 2001.

[KPW97] Seungjoo Kim, Sangjoon Park, and Dongho Won. Proxy signature, recisited. *In ICICS97*, 1334, 1997.

[LA03a] Sunder Lal and Amit Kumar Awasthi. Proxy blind signature scheme. *Cryptology ePrint Archive, Report 2003/072.*, 2003.

[LA03b] Sunder Lal and Amit Kumar Awasthi. A scheme for obtaining a warrant message from the digital proxy signatures. *Cryptology ePrint Archive, Report 2003/072.*, 2003.

[LCLS07] Huang Lin, Zhenfu Cao, Xiaohui Liang, and Jun Shao. Secure threshold multi authority attribute based encryption without a central authority. *Information Sciences*, 180(13):2618–1632, July 2007.

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. *In ACM Symposium on Principles of Distributed Computing*, pages 182–189, 2003.

[LK07] Jin Li and Kwangjo Kim. Attribute-based ring signature. *IACR ePrint*, 2007.

[LKK01] Byoungcheon Lee, Heesum Kim, and Kwangjo Kim. Secure mobile agent using strong non-designated proxy signature. *Information Security and Privacy-ACISP*, 2119:474–486, 2001.

[LKK03] Byoungcheon Lee, Heesun Kim, and Kwangjo Kim. Strong proxy signature and its applications. *The 2001 Symposium on Cryptography and Information Security (SCIS), LNCS*, pages 603–608, January 2003.

[LKPY06] Younho Lee, Heeyoul Kim, Yongsu Park, and Hyunsoo Yoon. A new proxy signature scheme providing self-delegation. *Information Security and Cryptology-ICISC, LNCS*, 4296:328–342, 2006.

[LM99] Sergio Loureiro and Refik Molva. Function hiding based on error correcting codes. *In Proceedings of the CryptTEC'99 International Workshop on Cryptographic Techniques and Electronic Commerce, M. Blum and C.Lee, Eds.*, pages 92–98, 1999.

[LW04] Chih-Yin Lin and Tzong-Chen Wu. An identity-based ring signature scheme from bilinear pairings. *the 18th International Conference on Advanced Information Networking and Applications, IEEE Computer Society*, 5927, 2004.

[McC90] Kevin McCurley. The discrete logarithm problem. *In Cryptology and computation number theory*, 42:49–74, 1990.

[MSS99] Paulo Jorge Marques, Luis Moura Sliva, and Joao Gabriel Silva. Security mechanisms for using mobile agents in electronic commerce). *Reliable Distributed Systems, Proceedings of the 18th IEEE Symposium*, pages 378–383, 1999.

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signature: Delegation of the power to sign message. *IEICE Trans. Fundamentals*, 79(9):1338–1353, 1996.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. crc press. *In Proc. of SAICSIT2002*, 1997.

[NAM05] Deholo Nali, Carlisle Adams, and Ali Miri. Using threshold attribute-based encryption for practical biometric-based access control. *International Journal of Network Security*, 1(3):173–182, Nov 2005.

[NNL01] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. *In J. Kilian, Editor, CRYPTO 2001, LNCS, Springer Verlag*, 2139:41–62, 2001.

[NT05] Samad Nasserian and Gene Tsudik. Revisiting oblivious signature-based envelopes. *In Financial Cryptography and Data Security*, 2005.

[Nwa96] Hyacinth S. Nwana. Software agents: an overview. *The knowledge engineering review*, 11:205–244, 1996.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. *In PKC2001, LNCS, Springer Verlag*, pages 104–118, 2001.

[PCC03] Stefan Poslad, Patricia Charlton, and Monique Calisti. Specifying standard security mechanisms in multi-agent systems. *TRUST,REPTATION, AND SECURITY: THEORIES AND PRACTICE, Lecture Notes in Computer Science*, 2631:227–237, 2003.

[PL01a] Hee-Un Park and Im-Yeong Lee. A digital nominative proxy signature scheme for mobile communication. *ICICS, LNCS*, 2229:451–455, 2001.

[PL01b] Hee-Un Park and Im-Yeong Lee. A digital nominative proxy signature scheme for mobile communication. *In ICICS 2001*, 2229, 2001.

[Rot01] Voller Roth. On the robustness of some cryptograpic protocols for mobile agent. *In Proceedings of the fifth International Conference on Mobile Agents. LNCS, Springer Verlag*, 2240:1–14, 2001.

[RS98] James Riordan and Bruce Schneier. Environmental key generation towards clueless agents. *In Mobile Agents and Security, G. Vigna, Ed., Lecture Notes in Computer Science, Springer-Verlag*, 1419:15–24, 1998.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. *ASIACRPT, LNCS*, 2248:552–565, 2001.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. *In Advances in Cryptology, Proc. CRYPTO84, LNCS*, 196:47–53, 1984.

[SL03] Seung-Hyun Seo and Sang-Ho Lee. A secure mobile agent system using multi-signature scheme in electronic commerce. *Web and communication Technologies and Internet-Related Social Issues, LNCS*, 2713:527–536, 2003.

[SLT99] H. M. Sun, N. Y. Lee, and T.Hwang. Threshold proxy signatures. *In IEEE Proceedings of Computers and Digital Techniques*, 146:259–263, 1999.

[ST98a] Tomas Sander and Christian F. Tschudin. On software protection via function hiding. *In Proceedings of the second International Workshop on Information Hiding, D. Aucsmith, Ed., Lecture Notes in Computer Science, Springer-Verlag*, pages 113–123, 1998.

[ST98b] Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. *Mobile Agent Security, LNCS*, 1419:44–60, 1998.

[Sun99] Sung-Min Sun. An efficient nonrepudiable threshold procy signature scheme with known signers. *Computer Communications*, 22(8):717–722, 1999.

[SW02] K. Shum and Victor K. Wei. A strong proxy signature scheme with proxy signer privacy protection. *In WET ICE'02*, 2002.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. *In Advances in Cryptology-Eurocrypt, LNCS, Springer*, 3494:457–473, 2005.

[TSM09a] Pairat Thorncharoensri, Willy Susilo, and Yi Mu. Policy-controlled signatures. *Information and Communications Security Lecture Notes in Computer Science, LNCS*, 5927:91–106, 2009.

[TSM09b] Pairat Thorncharoensri, Willy Susilo, and Yi Mu. Policy-controlled signatures and their applications. *Eleventh International Conference on Information and Communications Security (ICICS 2009)*, 5927:91–106, 2009.

[Vig97] Giovanni Vigna. Protecting mobile agents through tracing. *In proceedings of the Third ECOOP Workshop on Mobile Object Systems: Operating System Support for Mobile Object Systems*, 1997.

[Vig98] Giovanni Vigna. Cryptographic traces for mobile agents. *In Mobile Agents and Security, G. Vigna, Ed., Lecture Notes in Computer Science, Springer-Verlag*, 1419:25–43, 1998.

[VS98] Dennis Volpano and Geoffrey Smith. Language issues in mobile program security. *In Mobile Agents and Security, G. Vigna, Ed., Lecture Notes in Computer Science, Springer-Verlag*, pages 25–43, 1998.

[WBZD04] Guilin Wang, Feng Bao, Jianying Zhou, and Robert H. Deng. Security analysis of some proxy signatures. *ICISC2003, LNCS, Springer Verlag*, 2971:305–319, 2004.

[WLC98] Ching-Te Wang, Chu-Hsing Lin, and Chin-Chen Chang. Threshold signature schemes with traceable signers in group communications. *Computer Communiations*, 21(8):771–776, June 1998.

[Yee99] Bennet S. Yee. A sanctuary for mobile agents. *In Secure Internet Programming: Security Issues for Mobile and Distributed Objects, J. Vitek and C. Jensen, Ed., Lecture Notes in Computer Science, Springer-Verlag*, pages 261–274, 1999.

[Zha97] K. Zhang. Threshold proxy signature schemes. *In International Information Security Workshop*, 1997.

[ZK02] Fangguo Zhang and Kwangjo Kim. Id-based blind signature and ring signature from pairings. *Advances in Cryptology ASIACRYPT*, 2501:533–547, 2002.

[ZMZ09] Qi Zhang, Yi Mu, and Minjie Zhang. Secure mobile agent with designated hosts. *2009 Third International conference on Network and System Security, LNCS*, pages 286–293, 2009.

[ZSNL03] Fangguo Zhang, Reihaneh Safavi-Naini, and Chih-Yin Lin. New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairings. *Cryptology ePrint Archive, Report 2003/104.*, 2003.