SCIS 2001 The 2001 Symposium on Cryptography and Information Security
Oiso, Japan, January 23-26, 2001
The Institute of Electronics, Information and Communication Engineers

Secret Computation with Secrets for Mobile Agent using One-time Proxy Signature

Heesun Kim * sezsez@icu.ac.kr
Joonsang Baek † mohi@icu.ac.kr
Byoungcheon Lee * sultan@icu.ac.kr
Kwangjo Kim * kkj@icu.ac.kr

Abstract— As an application for electronic commerce, a mobile agent is now used to search for special products or services and is executed for a specific job designated by a customer in the server's environment on behalf of a customer. On the way of performing its role, a mobile agent can be vulnerable to several cryptographic attacks. These attacks can be more serious when done by malicious servers. Among schemes to resolve this problem, the concept of encrypted function for secret computation was proposed in [ST97, KBC00]. However, schemes that employ such encrypted functions enforce the server(host) to execute the functions of customer before verifying the mobile codes even in the case that the codes are maliciously modified. In this paper, we apply proxy signature scheme to the mobile agent system to enhance security and efficiency. Also, we suggest one-time proxy signature scheme to limit the signing power of the server.

Keywords: secret computation, proxy signature, one-time proxy signature

1 Introduction

1.1 Definition and Functionality of Mobile Agent

Mobile agents are autonomous software entities that are able to migrate across different execution environments. Mobility and autonomy make permanent connections unnecessary. Thus mobile agents are suitable for providing low-bandwidth connections and asynch- ronous communications. Furthermore, they provide better support for heterogeneous environments [KBC00].

The characteristics of mobile agents make them ideal for electronic commerce applications. A mobile agent can search for special products or services, negotiate with other entities and sell the products on behalf of a customer [KBC00].

We can consider an scenario that a mobile agent searches the price of the flight and books it. If the mobile agent finds the best condition presented by the server, the mobile agent digitally sign an order. Here, the mobile agent must carry the customer's private key and compute with the key, so that there are following fundamental problems [ST97] of executing mobile code:

1) Code and execution integrity : can a mobile agent protect itself against tampering by a malicious server?
2) Code privacy : can a mobile agent conceal the program it wants to have executed?
3) Computing with secrets in public : can a mobile agent remotely sign a document without disclosing the user's private key?

These problems require software based approach to be proven mathematically for the confidentiality and privacy. We will discuss about protecting mobile agents against malicious servers focusing on the software based cryptographic solution for the security issues of mobile agents.

* Information Security Group, Information and Communications Univ., 58-4 Hwaam-dong, Yusong-gu, Taejon, 305-732, Korea
† SECUi.COM, 647-9, Yeoksam-dong, Kangnam-gu Seoul, 135-080, Korea

1.2 Security Issues on Mobile Agent Paradigm

Considerable security issues in the mobile agent paradigm can be divided into the attacks by the server or by the mobile agent. In general, the former is shown harder to solve.

There are researches on detection and prevention of agent tampering in order to protect an agent from a malicious server. Detection of agent tampering includes mainly tracing identification of a server executing improper process and proving those improper execution. For the detection of agent tampering there are tracing mechanism to record operation of agent by Vigna [Vi98], agent service center by Yi *et al.* [YWL98], and multi-agent system by Kotzanikolaou *et al.* [KKC99].

The researches on prevention of agent tampering have been divided into the active and passive ones. Passive prevention has been studied on organization and structure of the system, mobile agent scheme executable in only trusted environment suggested by Farmer *et al.* [FGS96], and trading agent system implemented among the distributed entities suggested by Merwe and Sholms [MS97].

The researches on active prevention have been focused on protecting mobile agent without considering advantages of mobile agent, which is divided into the hardware and software based approach. Hardware-based approaches are less developed because of the high cost involved. On the other hand, software-based approaches are actively studied, which are an agent carrying time-limited token-data suggested by Hohl [Ho98] and privacy for the mobile code proposed by Loureiro and Molva [LM99]. And Sander and Tschudin designed a new digital signature scheme called the undetachable signature scheme based on CEF(Computing with Encrypted Function) for secure computation with secrets. In the undetachable signature scheme, a mobile agent can sign messages in the servers without revealing a signing secret key. Although they provided a concrete example of the undetachable signature scheme using algebraic homomorphic functions, it was shown to be insecure and subject to the Coppersmith, Stern and Vaudenay attack [CSV93]. More recently, Kotzanikolaou *et al.* proposed a way to construct a secure undetachable signature scheme based on RSA signature [KBC00].

We will prove that an agent guarantees the security of customer's secret key by transmitting the data (which can be opened) without transferring the signature function of the agent. For that proxy signature will be adapted regarding the secret computation of the secret key. Undetachable signature and proxy signature can be involved in such disputes requiring the priority policy among the order information [KBC00]. Moreover, we apply the one-time signature to minimize the disputes raised from the more than one valid signature of one server for more than one valid order information.

Organization : In Section 2, previous studies are reviewed by summarizing secret computation, then the application of the proxy signature scheme is suggested. In Section 3, we point out the limitation of the schemes in Section 2 and propose improving scheme. After the discussion of the security of the proposed scheme in Section 4, Section 5 concludes this paper.

2 Secret Computation with Secret Key in Public

In this section, we will describe the solution for secret computation with secret key in public. First, we will summarize the undetachable signature, and then, show application of the proxy signature to the mobile agent paradigm.

Notation
We will use the following notation throughout this paper:

- $hash$: hash function
- C : identification of a customer
- req_C : constraint of a customer (for examples, it includes description of goods, maximum price, delivery date, and issuing time of constraint, etc.)
- $h = hash(C, req\_C)$: binary strings computed with RSA modulus n
- S : identification of a server
- bid_S : bidding information of a server
- p, q : large primes, $q|p-1$
- α : generator for the subgroup G of Z_p^* having order q
- β : $\beta = \alpha^a \bmod p$, where a is a randomly chosen in $1 \leq a \leq q-1$ and kept secret by the trusted third party
- (x_C, y_C, y'_C) : customer's static private key and public key, $y_C = \alpha^{x_C} \bmod p, y'_C = \beta^{x_C} \bmod p$
- $(x_S, y_S) or (x_{S_i}, y_{S_j}, y'_{S_k})$: server's static private key and public key. $y_S = \alpha^{x_S} \bmod p$ for proxy signature. $y_{S_j} = \alpha^{x_{S_j}} \bmod p, y'_{S_k} = \beta^{x_{S_k}} \bmod p$, $s.t. j \in \{1, 3\}$, and $k \in \{2, 4,\}$ corresponding with $x_{S_i}, i \in \{1, 2, 3, 4,\}$ for one-time proxy signature.
- $msg = hash(S, C, req\_C, bid\_S)$: message to be signed
- m_w : warrant for the proxy signature
- $Sign(msg), Veri(msg)$: signature and verification schemes for message msg

2.1 Undetachable Signature Scheme

Undetachable signature scheme has been suggested to protect mobile agents against tampering attack or spying attack [KBC00, ST97]. The scheme suggests to carry the signature function in the mobile code and deliver the function to the server which makes it possible for the mobile agent not to expose the customer's private key and do some computation with the key. Sander and Tschudin [ST97] suggested undetachable signature scheme which is encrypted signature function based on CEF and is intended to solve the following problem :

> Alice has an algorithm to compute a function f. Bob has an input x and is willing to compute $f(x)$ for her, but Alice wants Bob to learn nothing substantial about f. Moreover, Bob should not need to interact with Alice during the computation of $f(x)$.

They achieved this concept using birational function. However, their scheme cannot guarantee the security and has weakness against Coppersmith, Stern and Vaudenay attack [CSV93]. On the other hand, Kotzanikolaou *et al* [KBC00] suggested undetachable signature scheme based on the security of RSA signature. Here, we will only describe undetachable signature scheme based on RSA signature in [KBC00].

Key Generation
For p and q, choose modulus n and publish it, where $n = pq$. Compute public key e of a customer and publish it, where $1 < e < \phi(n) = (p-1)(q-1)$, s.t. $gcd(e, \phi(n)) = 1$. Compute private key d of a customer, s.t. $1 < d < \phi(n), ed = 1 \bmod \phi(n)$.

Ready Phase of Mobile Agent
Customer defines encryption function f and encrypted signature function f_{signed} on input x as follows :

$$f(x) = h^x \bmod n$$
$$f_{signed}(x) = k^x \bmod n$$

Here, $k = h^d \bmod n$ is customer's RSA signature on h and $f_{signed}(x)$ is encrypted function of RSA signature function $s(x) = h^d \bmod n$. That is,

$$f_{signed}(x) = s \circ f(x) = s(f(x)) = s(h^x) = (h^x)^d = (h^d)^x = k^x.$$

Above $(f(x), f_{signed}(x))$ is included with (C, req_C) to the mobile code. Mobile agent drops by IP addresses of servers specified by a customer and finds the server which provides proper bidding information for the constraint of the customer.

Performance Phase of the Mobile Agent
Functions of mobile agent are executed on the input x by the server.

- $x = hash(S, C, bid\_S)$
- $m = f(x) = h^x \bmod n$
- $z = f_{signed}(x) = k^x \bmod n = (h^d)^x \bmod n = (h^x)^d \bmod n = m^d \bmod n = s(m)$

Server gains RSA signature (m, z). Mobile agent carries (S, C, bid_S), x, and (m, z) from the server to the customer.

Main features of this scheme are as follows [KBC00]:

F1 Bidding bid_S of the server which contradicts constraint req_C of the customer becomes invalid.

F2 Whole protocol executed by the mobile agent has asymmetric characteristics, since the server is not committed to the transaction whereas the customer is. Therefore, the signature of the server on the bidding bid_S is required to prevent from impersonation attack to the server.

F3 The scheme is described without considering confidentiality. However, mobile code from the customer is signed with the secret key of the customer and mobile code from the server is encrypted with the public key of the customer and signed with the secret key of the server to guarantee the confidentiality and integrity of the data.

F4 In purchasing one product, many server can generate each valid signature, (even, one server can generate more than one signature). To escape this problem, a priority policy needs to be set. The simplest way is that once the mobile agent signs it must return to the customer. Or the first generated signature must be recognized using the time-stamps. In spite of that, when the disputes occurs, participation of the agent broker or trusted third party is required.

On the other hand, this scheme has a few disadvantages.

D1 Server cannot recognize the modification or check validity of the mobile code until verifying the signature. So, the server must execute the signature function generated by the customer with validity check. That is, the server generates the signature with signature function and then must verify it with public key of the customer. If validity check is failed, it means mobile code has been modified or invalid. However, when hostile or malicious modified agent tries to attack the server, it may terminate the protocol without verification process after the generation of the signature by the transmitted signature function. Although protecting the server against malicious agent is out of the range of this paper, it is clear that this kind of attack is always the potential problem in the undetachable signature scheme.

D2 Customer's signature is generated with customer's secret key and is verified with customer's public key. Signature value has the information who is its owner but no information where it is executed. Because anyone who gets encrypted signature function can obtain customer's signature value. Therefore, server who executes signature functions must sign the signature of the customer with the secret key of the server to prevent impersonation attack. Later, the signature of the server on the signature of the customer will be required to confirm the validity of the signature of the customer.

2.2 Application of Proxy Signature Scheme

From the above scheme, we can list the requirements for computing with secret keys as follows :

Requirements

R1 The signature of the customer is generated by the server which is entrusted by the mobile agent.

R2 The security of transmitted messages against the impersonation attack, forgery attack, and etc. is guaranteed.

R3 Priority policy or trusted third party are required to solve the disputes which are from many valid bidding.

We will present more efficient scheme which satisfies the above requirements and overcomes the disadvantages of the undetachable signature. We can implement secret computation without revealing secret key by applying the proxy signature scheme whose architecture is similar with secret computation of customer's secret key by a mobile agent. In this section, we will show an example of the proxy signature scheme applied to the mobile agent.

2.2.1 Kim's Proxy Signature Scheme

We describe Kim's proxy signature scheme to use it to the mobile agent paradigm [Kim98].

1) Original signer C
   a) Chooses secret key $k_C \in_R Z_q^*$.
   b) Computes public key $r_C := \alpha^{k_C} \bmod p$.
   c) Computes $s_C := x_C \cdot hash(m_w, r_C) + k_C \bmod q$.
   c) Sends r_C, s_C, and m_w to S.
2) Proxy signer S
   a) Verifies $\alpha^{s_C} = y_C^{hash(m_w, r_C)} \cdot r_C \bmod p$.
   b) Computes proxy key $x_P := s_C + x_S \cdot hash(m_w, r_C)$ mod$p$.
   c) Computes public key $y_P := \alpha^{x_P} \bmod p$.
   d) Generates signature σ on the message msg with proxy key x_P, $\sigma = Sign(msg)$.
   e) Sends σ, and y_P to C.
3) Original signer C
   a) Checks the validity of the public key $y_P = (y_C \cdot y_S)^{hash(m_w, r_C)} \cdot r_C \bmod p$.
   b) Verifies σ with valid y_P, $msg = Veri(\sigma)$.

2.2.2 Application of Proxy Signature Scheme

Because of characteristics of the mobile agent paradigm, the customer cannot specify the server who executes the mobile code. Therefore, using strong non-designated proxy signature [LKK01], we designed one with emphasizing an customer's identification. To apply proxy signature, a customer generates delegation key pair and allows the server to sign with the key pair on behalf of the customer. Mobile agent carries the delegation key pair to the server. The server is specified by the mobile agent for the constraint of the customer.

1) Customer C performs the following at the local environment.
   a) Chooses private key $k_C \in_R Z_q^*$
   b) Computes r_C and s_C such as $r_C := \alpha^{k_C} \bmod p, s_C := x_C \cdot hash(C, req\_C, r_C) + k_C \bmod q$

   Customer carries the mobile agent with (C, r_C, s_C, req_C) to the network.
2) Mobile agent visits designated servers in the order of IP addresses. When bidding of a certain server is met for the constraint req_C of the customer, the mobile agent is executed by the server.

3) The following are executed in the remote environment.

   a) Server S verifies delegation key pair and computes message msg to sign bidding bid_S.

      $\alpha^{s_C} := y_C^{hash(C, req\_C, r_C)} \cdot r_C \bmod p$

      $msg = hash(S, C, bid\_S)$

   b) Server generates secret key x_P and public key y_P of the proxy key pair, signs the signature σ on msg, and delivers the results through the agent to the customer.

      $x_P := s_C + x_S \cdot hash(C, req\_C, r_C) \bmod q$

      $\sigma = Sign(msg)$

4) Customer C verifies transmitted messages from the server S.

   a) After computing $m = hash(S, C, bid\_S)$, checks if $m = msg$.

   b) Verifies public key y_P.

      $y_P := (y_C \cdot y_S)^{hash(C, req\_C, r_C)} \cdot r_C \bmod p.$

   c) Verifies signature σ for message msg with verified y_P.

2.3 Undetachable Signature Scheme vs. Proxy Signature Scheme

The comparison of the two schemes with respect to message size, detection phase of modified code, and meaning of signature is described in Table 1. In case of the undetachable signature scheme, the message size of the functions carried by the agent is $2|n|$. For the detection phase of modified code, a server can notice the modification of the mobile code at signature verification stage executing the verification of the signature after the signature stage. Here, the meaning of signature involves only customer.

Table 1: Comparison of the Undetachable Signature & Proxy Signature Schemes

Distinguishing Point	Undetachable Signature	Proxy Signature
Message size	$(f(x), f_{signed}(x))$: $2\|n\|$	(r_C, s_C) : $\|p\| + \|q\|$
Detecting of modified mobile code	At the signature verification stage after signing	At the delegation key verification stage of keys from the customer
Meaning of the signature	Only customer's signature	Customer and server's signature

In case of the application of the proxy signature scheme, message size of the delegation key pair carried by the agent is $|p| + |q|$. And a server can notice the modification of the mobile code at the delegation key verification stage. It doesn't need to execute invalid code because the server can process the signature stage only if validity check of the key is successful. Here the meaning of signature involves both customer and server. Therefore, the signature in this scheme overcomes impersonation and forgery attacks.

Our proposed scheme satisfies the requirements R1 and R2. The generated signature is signed with secret key of the entrusted server. In the scheme, impersonation and forgery attacks are prevented because the customer can confirm the validity of the signature generated by a server certifying the identification of the server and verifying the signature with public key of the server and that of the customer. And because validity check is processed before signing the message, the server does not need to generate invalid signature with invalid mobile code. However, the disputes from more than one valid signature are in potential state because the scheme also allow the server to have complete signing power. Therefore, this scheme requires the priority policy or participation of trusted third party like the undetachable signature scheme.

Here, we considered the way to minimize the disputes. We can recognize the signature generated by one server valid using the priority policy, which makes the mobile agent possible to return immediately to the customer after being first executed. And we may recognize one valid signature for one server with time-stamps. But actually, the security and correctness for the time-stamps are hard to be guaranteed because it is possible to be forged by the server. So, we thought the way allows one server to generate only one valid signature. We will describe the solution in the next section.

3 Secret Computation with Secrets using One-time Proxy Signature Scheme

We suggest new signature scheme to minimize possible disputes based on the priority policy. It is achieved by allowing one server to produce only one valid signature. We use one-time characteristic of fail-stop signature scheme to do so.

3.1 Fail-stop Signature Scheme

We use fail-stop signature [HP93] to limit power of the signature of proxy signer (server). Fail-stop signature is one-time signature to sign only one message with given key and provide high security against powerful third party. Fail-stop signature is as follows :

Key Generation Phase

1) Trustee T

   a) Computes p, q, and α.

   b) Chooses secret key $a \in_R Z_q^*$.

   c) Computes $\beta := \alpha^a \bmod p$, where a is kept secret.

   d) Publishes p, q, α, and β.

2) Signer A

   a) Chooses secret keys x_1, x_2, y_1, and $y_2 \in_R Z_q^*$.

   b) Computes β_1 and β_2 such as $\beta_1 = \alpha^{x_1}\beta^{x_2}$, $\beta_2 = \alpha^{y_1}\beta^{y_2} \bmod p$.

   c) Publishes β_1 and β_2.

Signature and Verification Protocol

1) Signature scheme

   a) Generates signatures $s_{1,m}$ and $s_{2,m}$ for the message m.

      $s_{1,m} = x_1 + my_1 \bmod q$, $s_{2,m} = x_2 + my_2 \bmod q$

2) Verification scheme

   a) Computes v_1 and v_2.

      $v_1 = \beta_1\beta_2^m \bmod p$, $v_2 = \alpha^{s_{1,m}}\beta^{s_{2,m}} \bmod p$

b) Accepts the signatures if $v_1 = v_2$.

If a signer signs two message m and m' with one proxy key pair as below, its secret keys are revealed.

$$s_{1,m} = x_1 + my_1 \bmod q, s_{2,m} = x_2 + my_2 \bmod q$$

$$s_{1,m'} = x_1 + m'y_1 \bmod q, s_{2,m'} = x_2 + m'y_2 \bmod q$$

That is, anyone can compute secret keys x_1, x_2, y_1, and y_2 of the proxy signer from the above equations. Therefore, the signer may not generate more than two different signatures with the same key pair.

3.2 One-time Proxy Signature Scheme

The descriptions below will explain how the proxy signature scheme gets the one-timeness. In brief, the one-time proxy signature scheme generates proxy signature on only one message with one proxy key pair.

We assume that agent broker and trusted third party generate public keys p, q, α, β, and secret key a. Here, Constraint req_C of the customer has a unique value because it includes time information.

Initialization Phase (Trustee T)

a) Generates p, q and α.

b) Chooses secret key $a \in_R Z_q^*$.

c) Computes $\beta := \alpha^a \bmod p$.

d) Publishes p, q, α, and β.

Signature Phase by the Generation of Proxy Key

1) Customer C executes the follows in local environment to generate delegation key pair $(r_{C_1}, r_{C_2}, r_{C_3}, r_{C_4}), (s_{C_1}, s_{C_2}, s_{C_3}, s_{C_4})$.

   a) Chooses secret keys $k_{C_1}, k_{C_2}, k_{C_3}$, and $k_{C_4} \in_R Z_q^*$.

   b) Computes $r_{C_1}, r_{C_2}, r_{C_3}$, and r_{C_4}, where $r_{C_i} := \alpha^{k_{C_i}} \bmod p, i \in \{1, 3\}$, $r_{C_j} := \beta^{k_{C_j}} \bmod p, j \in \{2, 4\}$.

   c) Computes $s_{C_1}, s_{C_2}, s_{C_3}$, and s_{C_4}, where $s_{C_i} := x_C \cdot hash(C, req\_C, r_{C_i}) + k_{C_i} \bmod p, i \in \{1, 2, 3, 4\}$.

   The customer loads $C, (r_{C_1}, r_{C_2}, r_{C_3}, r_{C_4}), (s_{C_1}, s_{C_2}, s_{C_3}, s_{C_4})$, and req_C to the mobile agent and lets the agent move around the network. Mobile agent visits the designated servers according to the IP addresses specified by the customer. Meeting a certain server which presents the proper bidding, the mobile agent is executed in the server environment.

2) Server S computes signed message $msg = hash(S, C, bid\_S)$ and generates the proxy keys using the delegation keys from the customer. And the server signs the message msg with the proxy keys and delivers the signature pair and other values through the agent.

   a) Verifies the delegation keys from the customer $\alpha^{s_{C_i}} := y_C{}^{hash(C, req\_C, r_{C_i})} \cdot r_{C_i}, i \in \{1, 3\}$, $\beta^{s_{C_j}} := y_C'{}^{hash(C, req\_C, r_{C_j})} \cdot r_{C_j}, j \in \{2, 4\}$. If the result of the verification is invalid, the server stops the protocol.

   b) Computes proxy keys $x_{P_1}, x_{P_2}, x_{P_3}$, and x_{P_4} to generate the signature.

   $x_{P_i} := s_{C_i} + x_{S_i} \cdot hash(C, req\_C, r_{C_i}), i \in \{1, 2, 3, 4\}$.

   c) Computes public key β_1, and β_2 for the proxy signature where, $\beta_1 = \alpha^{x_{P_1}} \beta^{x_{P_2}} \bmod p$, $\beta_2 = \alpha^{x_{P_3}} \beta^{x_{P_4}} \bmod p$

   d) Generates the signatures $\sigma_{1,m}$, and $\sigma_{2,m}$ for the message msg using proxy keys $x_{P_1}, x_{P_2}, x_{P_3}$, and x_{P_4}.

   $\sigma_{1,m} = x_{P_1} + msg \cdot x_{P_2} \bmod q, \sigma_{2,m} = x_{P_3} + msg \cdot x_{P_4} \bmod q$

   f) Server S delivers $(S, C, bid\_S), msg, (\beta_1, \beta_2)$, and $(\sigma_{1,m}, \sigma_{2,m})$ through the agent to the customer C.

3) Customer C verifies transmitted messages through the agent.

   a) Checks if $m = msg$ after computing $m = hash(S, C, req\_C, bid\_S)$.

   b) Verifies the validity of the public keys.

   $\beta_1 := (y_C \cdot y_{S_1})^{hash(C, req\_C, r_{C_1})}$
   $\cdot (y_C' \cdot y_{S_2})^{hash(C, req\_C, r_{C_2})} \cdot r_{C_1} \cdot r_{C_2} \bmod p$.

   $\beta_2 := (y_C \cdot y_{S_3})^{hash(C, req\_C, r_{C_3})}$
   $\cdot (y_C' \cdot y_{S_4})^{hash(C, req\_C, r_{C_4})} \cdot r_{C_3} \cdot r_{C_4} \bmod p$.

   c) Computes v_1 and v_2 to verify the signatures.

   $v_1 = \beta_1 \beta_2^{msg} \bmod p, v_2 = \alpha^{\sigma_{1,m}} \beta^{\sigma_{2,m}} \bmod p$

   Accepts the signature if $v_1 = v_2$.

One-timeness of the Signature

Server generates the proxy keys and may make more than two signatures with the keys. For example, when the server intends to sign the message msg and msg' ($msg = hash(S, C, req\_C, bid\_S), msg' = hash(S, C, req\_C, bid'\_S)$), signature $(\sigma_{1,m}, \sigma_{2,m})$ and $(\sigma_{1,m'}, \sigma_{2,m'})$ are

$$\sigma_{1,m} = x_{P_1} + msg \cdot x_{P_2} \bmod q,\ \sigma_{2,m} = x_{P_3} + msg \cdot x_{P_4} \bmod q$$

$$\sigma_{1,m'} = x_{P_1} + msg' \cdot x_{P_2} \bmod q,\ \sigma_{2,m'} = x_{P_3} + msg' \cdot x_{P_4} \bmod q$$

Here, two pairs of the signatures must be generated with the same proxy keys because proxy keys are able to be generated by the delegation key pair of the customer. Therefore, from the above two pairs of the signatures, proxy keys $(x_{P_1}, x_{P_2}, x_{P_3}, x_{P_4})$ are computed. And from the proxy keys, static secret keys $(x_{S_1}, x_{S_2}, x_{S_3}, x_{S_4})$ of the server be revealed. Therefore, server won't sign more than two messages with the same proxy key pair. And one-timeness of the signature is guaranteed.

4 Security Analysis

The securities of the proposed scheme are analyzed as follows :

Proposition 1 *The delegation key pair $r_{C_i}, s_{C_i}, i \in \{1, 2, 3, 4\}$ can be generated only by the customer C.*

Proof: The security of the delegation key pair generated by the customer C is based on the discrete logarithm problem. Therefore, anyone doesn't know customer's secret key cannot compute delegation key pair. The delegation key pair is verified with the public key of the customer, so that, it is secure against impersonation attack. □

Proposition 2 *The proxy key $x_{P_i}, i \in \{1, 2, 3, 4\}$ can be generated only by the server.*

Proof: The security of the server S is based on the discrete logarithm problem. The proxy keys $x_{P_1}, x_{P_2}, x_{P_3}$, and x_{P_4} generated by the server are blinded by the static secret keys $x_{S_1}, x_{S_2}, x_{S_3}$, and x_{S_4} of the server. Moreover, any server cannot make the delegation key pair of the customer from the **Proposition 1**. Therefore, only one who knows $x_{S_1}, x_{S_2}, x_{S_3}$, and x_{S_4} and receives the delegation key pair can generate the proxy keys. The proxy key pair is verified with the public keys of the customer and the server, so that, it is secure against impersonation and forgery attacks. □

Proposition 3 *The valid signature generated by the server using one-time proxy signature scheme can be generated only once.*

Proof: Constraint req_C has a unique value. Delegation key pair carried by the agent is unique because it has been determined by req_C. Proxy keys are generated only one pair because it has been determined by the delegation key pair. And server generates one signature pair with a proxy key pair as seen in the previous section, so that, the generation of only one valid signature is guaranteed. □

Proposition 4 *The security of the signature is based on that of the fail-stop signature.*

Proof: The security of the signature is explained with the probability for computing forgery of the signature [HP93]. To prove that a signature $\sigma' = (\sigma'_{1,m}, \sigma'_{2,m})$ on a message msg is a forgery, the signer derives the integer $a = log_\alpha \beta$ which serves as proof of forgery. The proxy signer should do the following :

(1) Compute a signature pair $\sigma = (\sigma_{1,m}, \sigma_{2,m})$ for message msg using its private keys x_{P_1}, x_{P_2}, x_{P_3}, and x_{P_4}.

(2) If $\sigma = \sigma'$ return to step (1). (Here, the probability that $\sigma = \sigma'$ is $1/q$.)

(3) Compute $a = (\sigma_{1,m} - \sigma'_{1,m}) \cdot (\sigma_{2,m} - \sigma'_{2,m})^{-1} \bmod q$. □

However, we have not considered the efficiency for the proposed scheme. Actually, length of messages transmitted by a mobile agent may carry a communication load. The efficiency as well as security for secret computation must be discussed further in the future study.

5 Conclusion

We have inspected closely security issues for the mobile agent paradigm applied to electronic commerce. Specially, among the researches on protecting mobile agents from the malicious server, we focused on the secret computation which is called undetachable signature scheme based on CEF. We analyzed this scheme and presented another example for secret computation using proxy signature scheme. Proposed scheme satisfies all features of the undetachable signature scheme as well as the validity check of the modified mobile code. Moreover, we proposed one-time proxy signature scheme to limit the signing power of the server by getting one-timeness.

We have focused on only secret computation of the secret key protecting a mobile agent. However, other various aspects of the security and the efficiency in the mobile agent paradigm need to be studied as the further works.

References

[CSV93] D. Coppersmith, J. Stern and S. Vaudenay, "Attacks on the Birational Permutation Signature Schemes", Proc. of Crypto'93, LNCS 773, Springer-Verlag, pp.435-443, 1993.

[FGS96] W. Farmer, J. Gutmann and V. Swarup, "Security for Mobile Agents: Authentication and State Appraisal", *Proc. of the European Symposium on Research in Computer Security (ESORICS)*, LNCS 1146, Springer-Verlag, pp.118-130, 1996.

[Ho98] F. Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts ", *Mobile Agent Security*, LNCS 1419, Springer-Verlag, pp.92-113, 1998.

[HP93] E.van Heyst, T.Pedersen, "How to Make Efficient Fail Stop Signatures", *In Advances in Eurocrypt'92*, LNCS 658, Springer Verlag, pp. 366–377, 1993.

[KBC00] P. Kotzanikolaou, M. Burmester and V. Chrissikopoulos, "Secure Transactions with Mobile Agents in Serverile Environments", *ACISP 2000*, LNCS 1841, Springer-Verlag, pp.289-297, 2000.

[Kim98] S. Kim, "Improved privacy and authenticity in digital signature/ key management", Ph.D. Thesis, SungKyunKwan University, 1998.

[KKC99] P. Kotzanikolaous, G. Katsirelos and V. Chrissikopoulos, "Mobile Agents for Secure Electronic Transactions", *Recent Advances in Signal Processing and Communications*, World Scientific and Engineering Society Press, pp.363-368, 1999.

[LKK01] Byoungcheon Lee, Heesun Kim and Kwangjo Kim, "Strong Proxy Signatures and Their Applications", Proc. of SCIS2001, Oiso, Japan, Jan. 23-26, 2001.

[LM99] S. Loureio and R. Molva, "Privacy for Mobile Code", *Proc. of Distributed Object Security Workshop OOPSLA'99*, 6 pages, 1999.

[MS97] J. Merwe and S.H. Solms, "Electronic Commerce with Secure Intelligent Trade Agents", *Proc. of ICICS'97*, LNCS 1334, Springer-Verlag, pp.452-462, 1997.

[ST97] T. Sander and C. F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts", *Mobile Agent Security*, LNCS 1419, Springer-Verlag, pp.44-60, 1997.

[Vi98] G. Vigna, "Cryptographic Traces for Mobile Agents", *Mobile Agent Security*, LNCS 1419, Springer-Verlag, pp.137-153, 1998.

[YWL98] Yi Xun, Wang Xiao Feng and Lam Kwok Yan, "A Secure Intelligent Trade Agent System", *Proc. of the International IFIP/GI Working Conference*, TREC'98, LNCS 1402, Springer-Verlag, pp.218-228, 1998.