Provably Secure Certificate-based Proxy Signature Schemes

Jiguo Li, Lizhong Xu, Yichen Zhang
College of Computer and Information Engineering, Hohai University, Nanjing, P.R.China, 210098
lijiguo@hhu.edu.cn, lzxu@hhu.edu.cn, zyc_718@163.com

***Abstract*—In this paper, we first propose the definition and security model of certificate-based proxy signature (CBPS). We then show that the certificate-based proxy signature scheme presented by Kang, Park and Hahn in CT-RSA 2004 are insecure against key replacement attacks. We further propose two certificate-based proxy signature schemes, which are shown to be existentially unforgeable against adaptive chosen message attacks under the computational Diffie-Hellman assumption in the random oracle model. Compared with the certificate-based proxy signature scheme in CT-RSA 2004, one of our schemes enjoys the same signature length and computation cost, another one requires a little more computation and communication cost.**

***Index Terms*-certificate-based signature; proxy signature; key replacement attack**

I. INTRODUCTION

In Eurocrypt 2003, Gentry [1] introduced the notion of certificate-based encryption (CBE). The main advantage of certificate-based encryption can be used to construct an efficient public-key infrastructure (PKI), solves certificate revocation problem and eliminate third-party queries in the traditional PKI. In addition, it also solves the inherent key escrow problem in the identity-based cryptography [2], [3]. The certificate-based encryption and signature have attracted a lot of attention since it is proposed. Yum and Lee [7] revisited the definitions and security notions of certificateless encryption (CL-PKE) and certificate-based encryption. They provided a formal equivalence theorem among identity-based encryption, certificateless encryption and certificate-based encryption. Galindo et al. [10] pointed out that a dishonest authority could break the security of the three generic constructions of CBE and CL-PKE schemes given in [7], [8]. These constructions were inherently flawed due to a naive use of double encryption as highlighted in [9]. Al-Riyami and Paterson [4] gave an analysis of Gentry's CBE concept and repaired a number of problems with the original definition and security model for CBE. They also provided a generic conversion showing that a secure CBE scheme could be constructed from any secure CL-PKE scheme. Kang and Park [6] pointed out that their conversion was incorrect due to the flaw in their security proof. This implies that the concrete CBE scheme by Al-Riyami and Paterson is therefore invalid. Recently, Lu et al [11] investigated the generic security of the CBE scheme obtained by applying the FO conversion to an arbitrary underlying OW-CBE-CPA secure CBE scheme and confirm that the FO conversion can generically convert any OW-CBE-CPA secure CBE into an IND-CBE-CCA secure CBE. They also note that the straightforward application of the FO conversion only leads to a CBE scheme with a loose reduction. They solved this problem by providing two security-enhancing conversions and achieved two generic CBE constructions [12], [13] from PKE and IBE, which are provably CCA-secure in the random oracle model. Lu et al [14] constructed an efficient CBE scheme with paring and prove it to be CCA-secure in the random oracle model based on the hardness of the Bilinear Diffie-Hellman Inversion problem. In parallel to CBE, Kang, Park and Hahn [5] proposed the security notion of certificate-based signature (CBS) that follows the idea of CBE presented by Gentry [1]. At the same time, they showed an application of CBS to proxy signatures [15], [16]. Li et al. [17] first introduced key replacement attack into the certificate-based system and refined the security model of certificate-based signature. They showed that the certificate-based signature scheme presented by Kang, Park and Hahn [5] was insecure against key replacement attacks. Furthermore they proposed a new secure and efficient certificate-based signature scheme, which was shown to be existentially unforgeable against adaptive chosen message attacks under the computational Diffie-Hellman assumption in the random oracle model. Au et al. [18] proposed a certificate-based (linkable) ring signature, which solved the problem of the complicated certificate chain verification in traditional PKI. Shao [20] presented a certificate-based verifiably encrypted signature from pairings and proved the new scheme was EUF-CMA secure in a stronger security model. Recently, Liu et al. [19] proposed two new certificate-based signature schemes with new features and advantages. The first one does not require any pairing computation, which is very efficient and its security can be proven using discrete logarithm assumption in the random oracle model. Another scheme can be proven secure in the standard model.

This work is supported by the National Natural Science Foundation of China (No. 60842002, 60673070), the National High-Tech Research and Development Plan of China under Grant (No.2007AA01Z409).

© 2009 ACADEMY PUBLISHER

Proxy signature is an important primitive to ensure the service availability issue. The concept of proxy signatures was first introduced by Mambo et al. [23] in 1996. A proxy signature scheme allows an entity to delegate signing capability to another entity in such a way that the latter can sign messages on behalf of the former when the former is not available. From a proxy signature, anyone can check both the original signer's delegation and the proxy signer's digital signature. Boldyreva et al. [24] formalized a notion of security for proxy signatures and showed that secure proxy signature schemes could be derived from secure standard signature schemes. Huang et al. [25] further refined the security model of the proxy signature and proposed some secure and efficient proxy signature schemes. Proxy signature schemes have attracted a considerable amount of interest from the cryptographic research community and have gained some research achievement [15,16,25,26].

In the paper, we first propose the formal definition and security model of certificate-based proxy signature. We then show that the certificate-based proxy signature scheme in [5] is insecure against key replacement attack. We further propose two certificate-based proxy signature schemes, analyze performance and the proposed schemes are proved secure in the random oracle model.

In the rest of the paper, it is organized as follows. In the next section, we review some preliminaries required in this paper. We describe the definition and security model of CBPS in Section III. In Section IV, we point out that CBPS scheme in [5] is insecure against key replacement attack. We propose two provably secure CBPS schemes and provide the security proof in Section V. In Section VI, we present a discussion on computation and communication efficiency. Finally, we conclude the paper in Section VII.

II. PRELIMINARIES

In this section, we review the knowledge about the bilinear pairing and computational Diffie-Hellman (CDH) problem.

Bilinear Pairing: Let G_1 denote an additive group of prime order p and G_2 be a multiplicative group of the same order. Let P be a generator of G_1 and $e: G_1 \times G_1 \rightarrow G_2$ be a bilinear mapping with the following properties:

- The map e is bilinear: $e(aP,bQ) = e(P,Q)^{ab}$ for all $P,Q \in G_1, a, b \in Z_p$.
- The map e is non-degenerate: $e(P,P) \neq 1 \in G_2$.
- The map e is efficiently computable.

CDH problem in G_1. Given (P,aP,bP), where $a, b \in Z_p^*$, compute abP. The success probability of any probabilistic polynomial-time algorithm A in solving the CDH problem in G_1 is defined to be

$$Succ_{A,G_1}^{CDH} = Pr[A(P, aP, bP) = abP : a, b \in Z_p^*].$$

The CDH assumption states that for every probabilistic polynomial-time algorithm A, $Succ_{A,G_1}^{CDH}$ is negligible.

III. CERTIFICATE-BASED PROXY SIGNATURE

In this section, we give the definition and security model of certificate-based proxy signature (CBPS) by integrating the idea of proxy signature [24-26] and certificate-based signature [17].

A. Definition of the Certificate-Based Proxy Signature

Let Alice denote the original signer and Bob the proxy signer. The certificate-based proxy signature consists of the following algorithms: **Setup, UserKeyGen, BLSSign, DelegationCertificateGen, ProxySign and Proxy Verification.**

1. **Setup**: This algorithm takes as input a security parameter 1^k and generates system parameter **params** and the original signer Alice secret/pubic key pair $(SK_A, PK_A) \in SK \times PK$. Here, SK denotes the set of the valid secret key values and PK denotes the set of the valid public key values. PK_A and **params** are shared in the system.
2. **UserKeyGen**: This algorithm takes as input system parameter **params**. It outputs a user ID's secret/pubic key pair $(SK_{ID}, PK_{ID}) \in SK \times PK$. Here, SK denotes the set of the valid secret key values and PK denotes the set of the valid public key values.
3. **BLSSign**(optional algorithm): This algorithm takes as input **params**, the signer's secret key SK_{ID} and the message m to be singed, this algorithm generates the standard signature: σ_S.
4. **DelegationCertificateGen**: This algorithm takes as input system parameter **params**, the original signer's secret key SK_A, the warrant w and proxy signer identity ID, this algorithm uses the **BLSSign** algorithm to generate the delegation certificate $Cert_{ID}$. Here, the warrant w consists of the proxy signer identity ID and pubic key PK_{ID}.
5. **ProxySign**: Input system's parameter **params**, the warrant w, the delegation certificate $Cert_{ID}$, the secret key SK_B of the proxy signer and the message m to be signed, this algorithm generates the proxy signature σ.
6. **ProxyVerification**: This algorithm takes as input a message/signature pair (m,σ), original signer's public keys PK_A, proxy signer's public key PK_B, the warrant w and system's parameter **params**. It outputs true if (m,σ) is valid. Otherwise, outputs false.

In **DelegationCertificateGen** algorithm, we regard the original signer as semi-trusted third party and integrate delegation algorithm of proxy signature and certificate generation algorithm of CBS into one **Delegation CertificateGen** algorithm, which makes CBPS scheme more efficient. The certificate generates by **Delegation CertificateGen** algorithm, that is, a short signature plays threefold role, firstly acts as the binding the public key of proxy signer and its holder, secondly acts as a partial signature key, and thirdly acts as delegation information

© 2009 ACADEMY PUBLISHER

about the warrant of proxy signer. We can use short signature algorithm [21, 22] to the above goal.

B. Security Models of CBPS

The first security model of proxy signature was proposed in [24]. Huang et al. [25, 26] further refined the security model of the proxy signature, which they divide the potential attackers into three kinds. Li et al. [17] first introduced key replacement attack into the certificate-based system and refined the security model of certificate-based signature. In this subsection, we follow the main idea in [17, 24-26] and divide the potential attackers into the following two kinds in essence.

Type I: This type adversary A_I not only has the public keys of Alice and Bob, he also has the secret key of the proxy signer Bob. For the adversary A_I, we mainly model the malicious proxy signer.

Type II: This type adversary A_{II} not only has the public keys of Alice and Bob, he also has the secret key of the original signer Alice. For the adversary A_{II}, we mainly model the malicious original signer.

Existential unforgeability against adaptive A_I

In this section, we consider the first kind of Type I adversary A_I. Informally, we want to capture the attack scenarios where an adversary wants to forge a valid CBPS under the warrant w^* and the public key PK_{ID^*} whose certificate and the delegation of this warrant are not known to him. The pubic key PK_{ID^*} might be the genuine one generated by the user ID^* or the fake one chosen by the adversary. A_I has the following capability:

1. A_I can obtain some message/signature pairs (m_i, σ_i) which are generated by the proxy signer ID.
2. A_I can replace the proxy signer ID^* public key with PK_{ID^*} which is chosen by himself. He can also dupe any other third party to verify user ID^*'s signatures using the false public key PK_{ID^*}.
3. If A_I has replaced the proxy signer ID^* public key, he cannot obtain the certificate of the false public key and the delegation of warrant w^* from the original signer.

The security of CBPS against a key replacement attack is defined by the following game between A_I and the challenger C:

- **Setup**: The challenger C runs the algorithm **Setup** and returns the original signer's pubic key PK_A and system parameter **params** to A_I.
- **UserKeyGen** queries: On a **UserKeyGen** query ID, if ID has already been created, nothing is to be carried out by C. Otherwise, C runs the algorithm **UserKeyGen** and obtains the secret/public key pair (SK_{ID}, PK_{ID}). Then it adds (ID, SK_{ID}, PK_{ID}) to the list *Key-List*. In this case, ID is said to be created. In both cases, PK_{ID} is returned.
- **Corruption queries**: On a **Corruption** query *ID*, where ID denotes the identity of proxy signer which has been created, C checks the list *Key-List* and returns the secret key SK_{ID}.
- **DelegationCertificateGen queries**: A_I can request query about the delegation on the warrant *w* and the certificate of proxy signer ID. In response, C runs the **DelegationCertificateGen** algorithm to obtain $Cert_{ID}$ and returns $Cert_{ID}$ to the adversary A_I. Note that $Cert_{ID}$ is the delegation on the warrant *w* and the certificate of the pair (ID, PK_{ID}) where PK_{ID} is the public key returned from the oracle **UserKeyGen**.
- **ProxySign queries**: A_I can request the proxy signature on the message *m* under the warrant *w* and proxy signer ID. In response, C runs **Delegation CertificateGen** algorithm to generate the delegation on the warrant *w* and certificate. Then C runs the **ProxySign** algorithm to obtain the proxy signature σ and returns σ to the adversary A_I.
- **Forge**: Finally, A_I outputs a signature σ^* with the warrant w^*, ID^* and the message m^* such that

1. σ^* is a valid CBPS on the message m^* under the public key PK_{ID^*} and the warrant w^*. Here, PK_{ID^*} is chosen by A_I and might not be the one returned from the oracle **UserKeyGen**.
2. w^* and ID^* has not been requested as one of the **DelegationCertificateGen** queries.
3. (m^*, w^*, ID^*) has not been requested as one of the **ProxySign** queries.

Compared with the security model defined in [5,24-26], an important refinement is that we allow the A_I to replace the target proxy signer's public key with any value chosen by him which captures the essence of the adversaries in the CBPS. However, A_I cannot obtain the delegation of the warrant and the certificate of the proxy signer's public key. In addition, we allow A_I to corrupt any proxy signer (except target proxy signer) in the system which is in order to reflect the malicious user who tries to only use his own secret key (without the knowledge of certificate and delegation) to generate valid signatures. The success probability of adaptively chosen message and chosen identity adversary A_I wins the above games is defined as $Succ_{A_I,CBPS}^{EF-CMA,CIDA}$. We say a CBPS scheme is existential unforgeable against a (t,q) chosen message and chosen identity adversary A_I, if A_I runs in polynomial time t, makes at most q queries and $Succ_{A_I,CBPS}^{EF-CMA,CIDA}$ is negligible.

Existential unforgeability against adaptive A_{II}

The existential unforgeability of a CBPS scheme under a type II attacker requires that it is difficult for the original signer to generate a valid proxy signature of a message m^* without the help of the proxy signer. A_{II} has the following capability:

© 2009 ACADEMY PUBLISHER

1. A_{II} has the knowledge of the original signer's secret key SK_A.
2. A_{II} can obtain some message/signature pairs (m_i, σ_i) which are generated by the proxy signer ID_i.
3. A_{II} cannot replace any proxy signer's public key.

The security of CBPS is defined by the following game between A_{II} and the challenger C:

- **Setup**: The challenger C runs the algorithm **Setup** and returns the original signer's pubic key PK_A and system parameter **params** to A_{II}.
- **UserKeyGen queries**: On a **UserKeyGen** query ID, if ID has already been created, nothing is to be carried out by C. Otherwise, C runs the algorithm **UserKeyGen** and obtains the secret/public key pair (SK_{ID}, PK_{ID}). Then it adds (ID, SK_{ID}, PK_{ID}) to the list *Key-List*. In this case, ID is said to be created. In both cases, PK_{ID} is returned.
- **Corruption queries**: On a **Corruption** query *ID*, where ID denotes the identity of proxy signer which has been created, C checks the list *Key-List* and returns the secret key SK_{ID} to A_{II}.
- **BLSSign queries** (optional query): Proceeding adaptively, A_{II} can request proxy signer's short signature on the message *m*. In response, C runs the **BLSSign** algorithm to generate the short signature on the message *m* and returns to the adversary A_{II}.
- **ProxySign queries**: A_{II} can request the proxy signature on the message *m* under the warrant *w* and proxy signer ID. In response, C runs **DelegationCertificateGen** algorithm to generate the delegation on the warrant *w* and certificate on (ID, PK_{ID}). C runs **UserKeyGen** algorithm to generate secret key SK_{ID} of proxy signer. Then C runs the **ProxySign** algorithm to obtain the proxy signature σ and returns σ to the adversary A_{II}.
- **Forge**: Finally, A_{II} outputs a signature σ^* with the warrant w^*, ID^* and the message m^* such that

1. σ^* is a valid CBPS on the message m^* under the public key PK_{ID^*} and the warrant w^*, where PK_{ID^*} is the public key output from the **UserKeyGen** query ID^*.
2. ID^* has never been submitted to corruption query.
3. (m^*, w^*, ID^*) has not been requested as one of the **ProxySign** queries.

For the second kind of Type II adversary A_{II}, he has the knowledge of the original signer's secret key SK_A. Therefore, A_{II} can not obtain secret key of the proxy signer. We adds a restriction condition, that is, ID^* has never been submitted to corruption query in the game between A_{II} and the challenger C, which remedies the deficiency about the security model of a certificated-based signature proposed in literature [17]. In addition, we also allow the attacker A_{II} can submit **BLSSign** queries, this is to guarantee that proxy signer's short signature on the message m^* can not help the attacker to forge a valid CBPS on the same message. The success probability of adaptively chosen message and chosen identity adversary A_{II} wins the above games is defined as $Succ_{A_{II},CBPS}^{EF-CMA,CIDA}$. We say a CBPS scheme is existential unforgeable against a (t,q) chosen message and chosen identity adversary A_{II}, if A_{II} runs in polynomial time t, makes at most q queries and $Succ_{A_{II},CBPS}^{EF-CMA,CIDA}$ is negligible.

IV. Key Replacement Attack for CBPS Scheme

Kang et al. [5] proposed a CBPS scheme and claimed that their scheme is secure under the security notion defined in [24]. We point out that their scheme is insecure against key replacement attack. In order to facilitate analysis,we first review the proxy signature scheme in [5].

A. Review of CBPS Scheme

We follow the denotations in [5]. Assume that there are two participants, Charlie and Alice with secret and public key pairs $(s_C, s_C P)$ and $(s_A, s_A P)$ respectively, and that they have the common system parameters $params = (G_1, G_2, e, P, H_1, H_3)$. The proxy signature algorithm works as follows:

S.Sign: A standard signature for message m is obtained by signing the result using **BLS.Sign**.

S.Vrfy: The verification of a signature σ for a message m is done by computing **BLS.Vrfy**.

(PS.Del, PS.Pro): In order to designate Alice as a proxy signer, Charlie simply sends to Alice an appropriate warrant w together with a signature $Cert_A = s_C P_A$, where $P_A = H_1(PK_C, PK_A, w)$. The corresponding proxy signing key of Alice is $SKP_A = Cert_A + s_A P_A$.

PS.Sign: A proxy signature for message m produced by Alice on behalf of Chalie, contains a warrant w, the public key of the proxy signer PK_A, and signature $\sigma = (U, V)$,where $U = rP_A$, $h = H_3(m, U)$ and $V = (r+h)SKP_A = (r+h)(s_C + s_A)P_A$.

PS.Vrfy: To verify a signature $(PK_C, m, (PK_A, w, \sigma))$, checks whether $e(PK_C + PK_A, U + hP_A) = e(P, V)$, where $h = H_3(m, U)$.

PS.Iden: The identification algorithm is defined as $PS.Iden(PK_A, w, \sigma) = PK_A$.

B. A Concrete Key Replacement Attack

We will show that the proxy scheme above is insecure against key replacement attack. The attack method is as follows.

The adversary first chooses a random value $r \in Z_q^*$, computes $PK'_A = rP - PK_C$ and replaces Alice's public key PK_A with PK'_A. Then the adversary chooses a

© 2009 ACADEMY PUBLISHER

random value $U \in G_1$, computes $P_A = H_1(PK_C, PK'_A, w)$, $h = H_3(m,U)$. Finally, the adversary computes $V = rU + rhP_A$. Thus, $\sigma = (U,V)$ is a valid certificate-based proxy signature. This is because signature $\sigma = (U,V)$ satisfies the following verification equation.

$$\begin{aligned} & e(PK_C + PK'_A, U + hP_A) \\ & = e(PK_C + rP - PK_C, U + hP_A) \\ & = e(P, rU + rhP_A) \\ & = e(P,V) \end{aligned}$$

V. PROXY SIGNATURE SCHEMES

In this section, we propose two provably secure certificate-based proxy signature schemes: one is denoted as the **CBPSm** proxy signature scheme, the other is denoted as **CBPSa** proxy signature scheme.

*A. **CBPSm** Proxy Signature Scheme*

In this subsection, we propose a provably secure proxy signature scheme. The proxy signature scheme is as follows:

1. **Setup:** Let G_1, G_2 be groups of a prime order p. $e: G_1 \times G_1 \to G_2$ is a bilinear pairing map. $H_1: \{0,1\}^* \times G_1 \times G_1 \to G_1$, $H_2: \{0,1\}^* \times G_1 \times G_1 \to G_1$ and $H_3: \{0,1\}^* \times \{0,1\}^* \times G_1 \times G_1 \to G_1$ are three secure cryptographic hash functions. $P \in G_1$ is an arbitrary generator of G_1.The original signer Alice selects a random number $s_A \in Z_p^*$ as her secret key and compute her public key $PK_A = s_A P \in G_1$. $params = < G_1, G_2, e, p, P, H_1, H_2, H_3 >$ are the system parameters. PK_A and $params$ are shared in the system.
2. **UserKeyGen:** Given $params$, proxy signer selects a random number $s_{ID} \in Z_p^*$ as his secret key and compute his public key $PK_{ID} = s_{ID} P \in G_1$.
3. **DelegationCertificateGen:** Given PK_A and $params$, Alice uses short signature algorithm to compute $Cert_{ID} = s_A P_A$ as the delegation certificate of proxy signer, where $P_A = H_1(w, PK_A, PK_{ID})$, the warrant w contains time stamp, proxy signer identity ID and public key PK_{ID} etc. The proxy signing key is $SKP_{ID} = (Cert_{ID}, s_{ID})$.
4. **ProxySign:** Given system parameter $params$, the warrant w, the delegation certificate $Cert_{ID}$, the secret key s_{ID} of the proxy signer and the message m to be signed, proxy signer selects a random number $r \in Z_p^*$ and computes **CBPS** $\sigma = (U,V)$, where $V = Cert_{ID} + s_{ID} H_2(m,U,PK_{ID}) + rH_3(m,w,U,PK_{ID})$ and $U = rP$.
5. **ProxyVerification:** This algorithm takes as input a message/signature pair (m,σ), original signer's public keys PK_A, proxy signer's public key PK_{ID}, the warrant w and system parameter $params$. The verifier checks whether
$$e(P,V) = e(PK_A, P_A) e(PK_{ID}, H_2(m,U,PK_{ID})) \cdot e(U, H_3(m,w,U,PK_{ID}))$$
If the equation holds, outputs *true*. Otherwise, outputs *false*.

Correctness.

The correctness of our scheme follows from the following fact:

$$\begin{aligned} & e(PK_A, P_A) e(PK_{ID}, H_2(m,U,PK_{ID})) \\ & e(U, H_3(m,w,U,PK_{ID})) \\ & = e(P, s_A P_A) e(P, s_{ID} H_2(m,U,PK_{ID})) \\ & e(P, rH_3(m,w,U,PK_{ID})) \\ & = e(P, Cert_{ID} + s_{ID} H_2(m,U,PK_{ID}) + \\ & rH_3(m,w,U,PK_{ID})) \\ & = e(P,V) \end{aligned}$$

B. Security Analysis

The **CBPSm** proxy signature scheme constructed above is existentially unforgeable against adaptive chosen message attacks under the computational Diffie-Hellman assumption in the random oracle model. The following theorem shows that our scheme is secure under the security notion in **III**

Theorem1. If there is a (*t*, *q*) Type I adaptively chosen message and chosen identity adversary A_I and win the game with probability $Succ_{A_I,CBPS}^{EF-CMA,CIDA}$, then there exists another algorithm B which can solve a random instance of Computational Diffie-Hellman problem in polynomial time with success probability $Succ_{B,G_1}^{CDH}$ = $\frac{1}{q+1}(1-\frac{1}{q+1})^q Succ_{A_I,CBPS}^{EF-CMA,CIDA}$.

Proof: Let P be the generator of G_1. Algorithm B is given P_1, $P_2 \in G_1$ where $P_1 = aP$, $P_2 = bP$, (P, P_1, P_2) is a random instance of the Computational Diffie-Hellman problem. Its goal is to compute abP. Algorithm B will simulate the oracles and interact with the forger A_I as described below. In the proof, we regard the hash functions as the random oracles. Algorithm B starts by setting the original public key $PK_A = P_1 = aP$, where P_1 is the input of the CDH problem. B sends (P, PK_A) to the algorithm A_I.

UserKeyGen queries: On a new **UserKeyGen** query ID_i, B chooses a random number $s_{ID_i} \in Z_p$ and sets $(SK_{ID_i}, PK_{ID_i}) = (s_{ID_i}, s_{ID_i} P)$. Then, he adds $(ID_i, SK_{ID_i}, PK_{ID_i})$ into the list $Key-List$ and returns PK_{ID_i} to A_I.

H_1: On a new H_1 query ζ_i, B first chooses a random number $coin_i \in \{0,1\}$ such that $Pr[coin_i = 1] = \delta$ where the value of δ will be determined later.

© 2009 ACADEMY PUBLISHER

1. If $coin_i = 1$, B chooses a random number $c_i \in Z_p$ and sets $H_1(\zeta_i) = c_i P + P_2$ where P_2 is another intput of CDH problem.
2. Else $coin_i = 0$, B chooses a random number $c_i \in Z_P$ and sets $H_1(\zeta_i) = c_i P$.

In both cases, B will add (ζ_i, c_i) into $H_1 - List$ and return $H_1(\zeta_i)$ to A_I.

H_2: On a new H_2 query θ_i, B chooses a random number $d_i \in Z_p$ and sets $H_2(\theta_i) = d_i P$. Then, he adds (θ_i, d_i) into $H_2 - List$ and returns $H_2(\theta_i)$ as the answer.

H_3: On a new H_3 query ξ_i, B chooses a random number $\lambda_i \in Z_P$ and sets $H_3(\theta_i) = \lambda_i P$. Then, he adds (ξ_i, λ_i) into $H_3 - List$ and returns $H_3(\theta_i)$ as the answer.

DelegationCertificateGen queries: On delegation of the warrant w_i and certificate query ID_i, B first checks the list $Key - List$ to obtain this user's public key PK_{ID_i}. We assume that $(w_i, PK_A, PK_{ID_i}, .)$ has been in $H_1 - List$. Otherwise, B can add $(w_i, PK_A, PK_{ID_i}, c_i)$ into $H_1 - List$ as the same way he responds to H_1 queries.

1. If $coin_i = 0$, which means $P_A = c_i P = H_1(w_i, PK_A, PK_{ID_i})$, B returns the certificate $Cert_{ID_i} = c_i P_1$.
2. Otherwise, B aborts.

Corruption queries: On a corruption query ID_i, B will check the list $Key - List$ and return SK_{ID_i} to A_I.

ProxySign queries: On a sign query (m_i, ID_j), B first checks $H_1 - List$ to obtain (w_i, PK_A, c_j). If $coin_i = 0$, B can generate the certificate $Cert_{ID_j}$ as he responds the **DelegationCertificateGen** queries and use $(Cert_{ID_j}, SK_{ID_j})$ to sign the message m_i. Otherwise, $coin_j = 1$ and $H_1(w_i, PK_A, PK_{ID_i}) = c_j P + P_2$. Then, B chooses a random number $r_i \in Z_P$ and sets $U_i = r_i P - P_1$.

1. Firstly, he checks $H_2 - List$. If $(m_i, U_i, PK_{ID_j}, .)$ does not exist in $H_2 - List$, B will add $(m_i, U_i, PK_{ID_j}, d_i)$ into $H_2 - List$ as the same way he responds to H_2 queries.
2. Secondly, He checks whether $(m_i, w_j, U_i, PK_{ID_j}, .)$ exists $H_3 - List$. If it does, B must rechoose the number r_i until there is no collision. B further sets $H_3(m_i, w_j, U_i, PK_{ID_j}) = \lambda_i P + P_2$ and adds $(m_i, w_j, U_i, PK_{ID_j}, \lambda_i, \lambda_i P + P_2)$ into $H_3 - List$.
3. At last, B outputs (U_i, V_i) as the signature, where $V_i = c_j P_1 + s_{ID_i} H_2(m_i, U_i, PK_{ID_j}) + \lambda_i U_i + r_i P_2$.

Correctness

$$
\begin{aligned}
& e(V_i, P) \\
&= e(c_j P_1 + s_{ID_i} H_2(m_i, U_i, PK_{ID_j}) + \lambda_i U_i + r_i P_2, P) \\
&= e(c_j P_1 + abP + s_{ID_i} H_2(m_i, U_i, PK_{ID_j}) \\
&\quad + \lambda_i U_i + r_i P_2 - abP, P) \\
&= e(a(c_j P + P_2) + s_{ID_i} H_2(m_i, U_i, PK_{ID_j}) \\
&\quad + (r_i - a) H_3((m_i, w_j, U_i, PK_{ID_j}), P) \\
&= e(PK_A, P_A) e(H_2(m_i, U_i, PK_{ID_j}), PK_{ID_j}) \\
&\quad e(H_3((m_i, w_j, U_i, PK_{ID_j}), U_i)
\end{aligned}
$$

Finally, A_I outputs a valid forgery $(m^*, \sigma^* = (U^*, V^*))$ with probability $Succ_{A_I}^{cma,cida}$. Here, PK_{ID^*} is chosen by A_I and might not be ID^*'s public key output from the oracle **UserKeyGen**. We assume that $(w^*, PK_A, PK_{ID^*}, c^*)$, $(m^*, U^*, PK_{ID^*}, d^*)$, $(m^*, w^*, U^*, PK_{ID^*}, \lambda^*, \lambda^* P)$ have been in $H_1 - List$, $H_2 - List$ and $H_3 - List$ respectively. If (U^*, V^*) is a valid signature of the message m^*, then

$$V^* = aH_1(w^*, PK_A, PK_{ID^*}) + d^* PK_{ID^*} + \lambda^* U^*.$$

1. If $c^* = 1$, $H_1(w^*, PK_A, PK_{ID^*}) = c^* P + P_2$. Therefore, B can compute $abP = V^* - (c^* P_1 + d^* PK_{ID^*} + \lambda^* U^*)$.
2. Otherwise, B fails to solve this instance of CDH problem.

According to the simulation, B can compute the value of abP if and only if all the following three events happen:

E_1: B does not fail during the simulation.

E_2: A_I output a valid forgery.

E_3: In the forgery output by A_I, $c^* = 1$.

Therefore, the probability that B can solve this instance of CDH problem is $Succ_{B,G_1}^{CDH} = Pr[E_1 \wedge E_2 \wedge E_3] = Pr[E_1] Pr[E_2 | E_1] Pr[E_3 | E_1 \wedge E_2]$. In addition, all the simulation can be done in polynomial time.

From the simulation, we have $Pr[E_1] \geq (1-\delta)^q$, $Pr[E_2 | E_1] = Succ_{A_I, CBPS}^{EF-CMA,CIDA}$ and $Pr[E_3 | E_1 \wedge E_2] = \delta$. Thus, $Succ_{B,G_1}^{CDH} \geq \delta(1-\delta)^q Succ_{A_I, CBPS}^{EF-CMA,CIDA}$.

When $\delta = 1/(q+1)$, this probability is maximized at

$$Succ_{B,G_1}^{CDH} = \frac{1}{q+1}(1 - \frac{1}{q+1})^q Succ_{A_I, CBPS}^{EF-CMA,CIDA}.$$

Theorem2. If there is a (t, q) Type II adaptively chosen message and chosen identity adversary A_{II} and win the game with probability $Succ_{A_{II}, CBPS}^{EF-CMA,CIDA}$, then there exists another algorithm B which can solve a random instance of Computational Diffie-Hellman problem in polynomial time with success probability $Succ_{B,G_1}^{CDH} \geq \frac{1}{q'}(1 - \frac{1}{q'})^q Succ_{A_{II}, CBPS}^{EF-CMA,CIDA}$, where $1 \neq q' \leq q$ denotes the number of queries submitted to the oracle **UserKeyGen**.

© 2009 ACADEMY PUBLISHER

Proof: Let P be the generator of G_1. Algorithm B is given $P_1, P_2 \in G_1$, where $P_1 = aP, P_2 = bP$, (P_1, P_2, P_3) is a random instance of the Computational Diffie-Hellman problem. Its goal is to compute abP. Algorithm B will simulate the oracles and interact with the forger A_{II} as described below. In the proof, we regard the hash functions as the random oracles. Algorithm B starts by choosing a random number $s \in Z_p$ and sets $SK_A = s$, $PK_A = sP$. Then B sends (P, s, sP) to A_{II}.

UserKeyGen queries: On a new **UserKeyGen** query ID_i, B acts as following. Suppose there are up to q' **UserKeyGen** queries, B will choose a random number $\pi \in \{1,2,\cdots,q'\}$.

1. ID_i is the πth query, B sets $s_{ID_i} = \perp$ and $PK_{ID_i} = P_1$ where P_1 is the input of CDH problem. Here, the symbol $\perp$ means B doesn't know the corresponding value.
2. Otherwise B chooses a random number $s_{ID_i} \in Z_p$ and sets $PK_{ID_i} = s_{ID_i} P$.

In both cases, B adds $(ID_i, s_{ID_i}, PK_{ID_i})$ into the list *Key−List* and returns PK_{ID_i} to A_{II}.

H_1: On a new H_1 query ς_i, B chooses a random number $c_i \in Z_p$ and sets $H_1(\varsigma_i) = c_i P$. Then, B adds (ς_i, c_i) into H_1-*List* and return $H_1(\varsigma_i)$ to A_{II}.

H_2: On a new H_2 query θ_i, B chooses a random number $d_i \in Z_p$ and sets $H_2(\theta_i) = d_i P + P_2$. Then, he adds $(\theta_i, d_i, d_i P + P_2)$ into H_2-*List* and returns $H_2(\theta_i)$ as the answer.

H_3: On a new H_3 query ξ_i, B chooses a random number $\lambda_i \in Z_p$ and sets $H_3(\theta_i) = \lambda_i P$. Then, he adds $(\xi_i, \lambda_i, \lambda_i P)$ into H_3-*List* and returns $H_3(\theta_i)$ as the answer.

Corruption queries: On a corruption query ID_i, B will check the list *Key−List* and return s_{ID_i} to A_{II}. If $s_{ID_i} = \perp$, B fails to solve this problem.

ProxySign queries: On a sign query (m_i, ID_j), B first checks the list *L*.

1. If $s_{ID_j} = \perp$, B will choose two random elements: $U_i = r_i P \in G_1$ and $d_i \in Z_p$. Then, he adds $(m_i, U_i, PK_{ID_j}, d_i P)$ into H_2-*List*. If a collision occurs, U_i and d_i will be re-chosen. In addition, B will add $(m_i, w_j, U_i, PK_{ID_j}, \lambda_i, \lambda_i P)$ to H_3-*List* as the same way he responds to H_3 queries. By assumption, $(w_j, PK_A, PK_{ID_j}, c_j)$ has been in H_1-*List*. Then B computes $V_i = Cert_{ID_j} + d_i PK_{ID_j} + \lambda_i U_i$. The signature (U_i, V_i) is returned.

Correctness

$$\begin{aligned} e(V_i, P) &= e(Cert_{ID_j} + d_i PK_{ID_j} + \lambda_i U_i, P) \\ &= e(Cert_{ID_j} + s_{ID_j}(d_i)P + r_i(\lambda_i)P, P) \\ &= e(Cert_{ID_j}, P) e(s_{ID_j}(d_i)P, P) e(r_i(\lambda_i)P, P) \\ &= e(PK_A, H_1(w, PK_A, PK_{ID_j})) \\ & e(H_2(m_i, U_i, PK_{ID_j}), PK_{ID_j}) \\ & e(H_3(m_i, w_j, U_i, PK_{ID_j}, U_i) \end{aligned}$$

2. Otherwise, B can use s_{ID_j} and $Cert_{ID_j}$ to generate the signature on this message.

Finally, A_{II} outputs a valid forgery $(m^*, \sigma^* = (U^*, V^*))$ with probability $Succ_{A_{II},CBPS}^{EF-CMA,CIDA}$. We assume that $(w^*, PK_A, PK_{ID^*}, c^*)$, $(m^*, U^*, PK_{ID^*}, d^*, d^* P_2)$, $(m^*, w^*, U^*, PK_{ID^*}, \lambda^*, \lambda^* P)$ have been in H_1-*List*, H_2-*List and* H_3-*List* respectively. Here PK_{ID^*} is the public key of user ID^* output from the oracle **UserKeyGen**. If (U^*, V^*) is a valid signature of the message m^*, then $V^* = sc^* P + s_{ID^*}(d^* P + P_2) + \lambda^* U^*$.

1. If $PK_{ID^*} = P_1$, then s_{ID^*} should be a. Therefore, B can compute $abP = V^* - (sc^* P + d^* P_1 + \lambda^* U^*)$.
2. Otherwise, B fails to solve this instance of CDH problem.

According to the simulation, B can compute the value of abP if and only if all the following three events happen:

E_1: B does not fail during the simulation.

E_2: A_{II} output a valid forgery.

E_3: In the forgery output by A_{II}, $PK_{ID^*} = P_1$.

Therefore, the probability that B can solve this instance of CDH problem is

$$Succ_{B,G_1}^{CDH} = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1]\Pr[E_2 | E_1]\Pr[E_3 | E_1 \wedge E_2].$$

In addition, all the simulation can be done in polynomial time.

From the simulation, we have

$$\Pr[E_1] \geq (1 - \frac{1}{q'})^q,$$

$$\Pr[E_2 | E_1] = Succ_{A_{II},CBPS}^{EF-CMA,CIDA},$$

$$\Pr[E_3 | E_1 \wedge E_2] = \frac{1}{q'}.$$

Thus, $Succ_{B,G_1}^{CDH} \geq \frac{1}{q'}(1 - \frac{1}{q'})^q Succ_{A_{II},CBPS}^{EF-CMA,CIDA}$ where $1 \neq q' \leq q$ denotes the number of queries submitted to the oracle **UserKeyGen**.

*C. **CBPSa** Proxy Signature Scheme*

In this subsection, we propose a provably secure proxy signature scheme, which is based on the **CBSa** in [5].

© 2009 ACADEMY PUBLISHER

1. **Setup**: Let G_1, G_2 be groups of a prime order p in which there exists a bilinear pairing map $e: G_1 \times G_1 \to G_2$. $H_1: \{0,1\}^* \times G_1 \times G_1 \to G_1$, $H_2: \{0,1\}^* \to G_1$ and $H_3: \{0,1\}^* \times G_1 \times G_1 \to Z_p^*$ are three secure cryptographic hash functions. $P \in G_1$ is an arbitrary generator of G_1. The original signer Alice selects a random number $s_A \in Z_p^*$ as her secret key and compute her public key $PK_A = s_A P \in G_1$. The system parameters are $\text{params} =< G_1, G_2, e, p, P, H_1, H_2, H_3 >$. PK_A and params are shared in the system.
2. **UserKeyGen**: Given params, proxy signer selects a random number $s_{ID} \in Z_p^*$ as his secret key and compute his public key $PK_{ID} = s_{ID} P \in G_1$.
3. **DelegationCertificateGen**: Given params and PK_A, Alice uses short signature algorithm to compute $Cert_{ID} = s_A P_A$ as the delegation certificate of proxy signer, where $P_A = H_1(w, PK_A, PK_{ID})$, the warrant w contains time stamp, proxy signer identity ID and public key PK_{ID} etc. The corresponding proxy signing key is $SKP_{ID} = (Cert_{ID}, s_{ID})$.
4. **ProxySign**: Given system parameter params, the warrant w, the delegation certificate $Cert_{ID}$, the secret key s_{ID} of the proxy signer and the message m to be signed, proxy signer selects a random numberr $r \in Z_p^*$ and computes **CBPS** $\sigma = (U_1, U_2, V)$, where $U_1 = rP_A$, $U_2 = rP_A'$, $P_A' = H_2(w)$, $h = H_3(m, U_1, U_2)$ and $V = (r+h)(Cert_{ID} + s_{ID} P_A')$.
5. **ProxyVerification**: This algorithm takes as input a message/signature pair (m, σ), original signer's public keys PK_A, proxy signer's public key PK_{ID}, the warrant w and system parameter params. The verifier checks whether
$e(P,V) = e(PK_A, U_1 + hP_A) e(PK_{ID}, U_2 + hP_A')$.
If the equation holds, outputs true. Otherwise, outputs false.

Correctness.

The correctness of our scheme follows from the following fact:

$$\begin{aligned} & e(PK_A, U_1 + hP_A) e(PK_{ID}, U_2 + hP_A') \\ &= e(s_A P, rP_A + hP_A) e(s_{ID} P, rP_A' + hP_A') \\ &= e(P, (r+h) s_A P_A) e(P, (r+h) s_{ID} P_A') \\ &= e(P, (r+h)(Cert_{ID} + s_{ID} P_A')) \\ &= e(P, V) \end{aligned}$$

The **CBPSa** proxy signature scheme constructed above is existentially unforgeable against adaptive chosen message attacks under the computational Diffie-Hellman assumption in the random oracle model. Due to the limitation of the space, security proof of **CBPSa** scheme refers reader to that of **CBPSm** scheme.

VI. Efficiency Comparison

In this section, we first give the definition of the following notations. $|G_1|$ denotes bit length of an element in G_1, E denotes exponentiation in G_1, BA denotes bilinear pairing operation and PA denotes point addition in G_1. The following table shows the comparison of our scheme and the proxy signature scheme in [5].

TABLEI
COMPARISON OF PERFORMANCE EFFICIENCY

Scheme	Size	Signature	Verification
Scheme in [5]	$2\|G_1\|$	$2E$	$E + 2PA + 2BA$
CBPSm scheme	$2\|G_1\|$	$3E + 2PA$	$3BA$
CBPSa scheme	$3\|G_1\|$	$3E$	$2E + 2PA + 3BA$

As shown in the table, **CBSPm** proxy signature scheme enjoys the same signature length and computation cost as the scheme in [5]. **CBPSa** proxy signature scheme consists of 3 elements in G_1 and is about 170 bits longer than the proxy scheme in [5] when some suitable elliptic curve is used as the underlying building block. **CBPSa** proxy signature scheme also requires more operation cost than the proxy scheme in [5].

VII. Conclusion

In this paper, we propose the definition and security model of certificate-based proxy signature. Our analysis showed that **CBPS** scheme proposed by Kang, Park and Hahn is insecure against the key replacement attack. Furthermore, we constructed two certificate-based proxy signature schemes. Our proposal is proven existentially unforgeable against adaptive chosen message attacks in the random oracle model. The security depends on merely well known computational Diffie-Hellman assumption. Compared with the certificate-based proxy signature scheme in [5], **CBPSm** scheme enjoys the same signature length and computation cost, while **CBPSa** scheme is not as efficient as the scheme in [5]. Due to a merit of **CBS**, our **CBPS** scheme does not require a secure channel for proxy designation.

References

[1] Gentry, C.: "Certificate-based Encryption and the Certificate Revocation Problem," In: Biham, E. (ed.): Advances in Cryptology-Eurorypt'03. Lecture Notes in Computer Science, Vol. 2656. Springer-Verlag, Berlin Heidelberg New York (2003) 272–293

[2] Shamir, A.: "Identity-based Cryptosystems and Signature Schemes," In: Blakley, G.R., Chaum, D. (eds.): Advances in Cryptology-Crypto'84. Lecture Notes in Computer Science, Vol. 196. Springer-Verlag, Berlin Heidelberg New York (1985) 47–53

[3] Boneh, D., Franklin, M.: "Identity-based Encryption from the Weil Pairing," SIAM J. Comput. 32(2003) 586–615.

[4] Al-Riyami, S.S., Paterson, K.G.: "CBE from CL-PKE: A Generic Construction and Efficient Schemes," In:

© 2009 ACADEMY PUBLISHER

Vaudenay, S. (ed.): PKC'05. Lecture Notes in Computer Science, Vol. 3386. Springer-Verlag, Berlin Heidelberg New York (2005) 398–415

[5] Kang, B.G., Park, J.H., Hahn, S.G.: "A Certificate-based Signature Scheme," In: Okamoto, T. (ed.): CT-RSA'04. Lecture Notes in Computer Science, Vol. 2964. Springer-Verlag, Berlin Heidelberg New York (2004) 99–111

[6] Kang, B.G., Park, J.H.: "Is It Possible to Have CBE from CLPKE?" Cryptology ePrint Archive, Report 2005/431

[7] Yum, D.H., Lee, P.J.: "Identity-based Cryptography in Public Key Management," In: Katsikas, S.K. et al. (eds.): EuroPKI 2004. Lecture Notes in Computer Science, Vol. 3093. Springer-Verlag, Berlin Heidelberg New York (2004) 71–84

[8] Yum, D.H., Lee, P.J.: "Generic Construction of Certificateless Encryption," In: Lagana, A. et al. (eds.): Computational Science and Its Applications-ICCSA' 2004. Lecture Notes in Computer Science, Vol. 3043. Springer-Verlag, Berlin Heidelberg New York (2004) 802–811

[9] Dodis, Y., Katz, J.: "Chosen-Ciphertext Security of Multiple Encryption," In: Kilian, J. (ed.): Theory of Cryptography Conference-TCC'05. Lecture Notes in Computer Science, Vol. 3378. Springer-Verlag, Berlin Heidelberg New York (2005) 188–209

[10] Galindo, D., Morillo, P., R`afols, C.: "Breaking Yum and Lee Generic Constructions of Certificate-Less and Certificate-Based Encryption Schemes," In: Andrea S. Atzeni, Antonio Lioy (eds.): EuroPKI'2006. Lecture Notes in Computer Science, Vol. 4043. Springer-Verlag, Berlin Heidelberg New York (2006) 81–91

[11] Lu Yang, Li Jiguo and Xiao Junmo. "Applying the Fujisaki-Okamoto Conversion to Certificate-based Encryption," In the 2008 International Symposium on Electronic Commerce and Security, Guangzhou, China, pp. 296–300, 2008

[12] Lu Yang and Li Jiguo. "A General and Secure Certificate-based Encryption Construction," In the 3rd ChinaGrid Annual Conference, Dunhuang, China, pp. 182–189, 2008

[13] Lu Yang, Li Jiguo and Xiao Junmo. "Generic Construction of Certificate-based Encryption," In the 9th International Conference for Young Computer Scientists, Zhangjajie, China, 2008, 1589–1594

[14] Yang Lu, Jiguo Li, and Junmo Xiao. "Constructing Efficient Certificate-based Encryption with Paring," Journal of Computers. 2009, 4(1): 19–26

[15] Li Jiguo, Cao Zhenfu and Zhang Yichen. "Nonrepudiable Proxy Multi-Signature Schemes," Journal of Computer Science and Technology. 2003, 18(3): 399–402

[16] Li Jiguo, Liang Zhenghe, Zhu Yuelong and Zhang Yichen. "Improvement of Some Proxy Signature Schemes," Chinese Journal of Electronics, 2005, 14(3): 407–411

[17] Li, J. G., Huang, X. Y., Mu, Y., Susilo, W., Wu, Q. H.: "Certificate-Based Signature: Security Model and Efficient Construction," In: Lopez, J., Samarati, P., Ferrer, J.L. (eds.): EuroPKI' 2007, Lecture Notes in Computer Science, Vol. 4582, Springer-Verlag, Berlin, (2007)110–125

[18] Au, M. H., Liu, J. K., Susilo, W., Yuen, T. H.: "Certificate Based (Linkable) Ring Signature," In: Dawson, E., Wong, D. S. (eds.): ISPEC'2007, Lecture Notes in Computer Science, Vol. 4464, Springer-Verlag, Berlin, (2007) 79–92

[19] Joseph K. Liu, Joonsang Baek, Willy Susilo, and Jianying Zhou. "Certificate-Based Signature Schemes without Pairings or Random Oracles," In: Wu, T.-C. et al. (Eds.): ISC'2008, LNCS 5222, pp. 285–297, 2008.

[20] Zuhua Shao. "Certificate-based verifiably encrypted signatures from pairings," Information Sciences, 178 (2008) 2360–2373.

[21] F. Zhang, R. Safavi-Naini, and W. Susilo. "An efficient signature scheme from bilinear pairings and its applications," In Public Key Cryptography (PKC'04), LNCS 2947, pages 277–290. Springer-Verlag, 2004

[22] Boneh, D., Lynn, B., and Shacham, H. "Short signatures from the Weil pairing," Advances in Cryptology-ASIACRYPT 2001, C. Boyd (Ed.), Lecture Notes in Comput. Sci. 2248, Springer-Verlag, pp. 514–532 (2001)

[23] Mambo, M., Usuda, K., Okamoto, E. "Proxy signatures for delegating signing operation," In Proc. 3rd ACM Conference on Computer and Communications Security, ACM Press, 1996, 48–57

[24] Boldyreva, A., Palacio, A., and Warinschi, B. "Secure proxy signature schemes for delegation of signing rights," Cryptology ePrint Archive, Report 2003/096

[25] Xinyi Huang, Yi Mu, Willy Susilo, Fangguo Zhang and Xiaofeng Chen. "A Short Proxy Signature Scheme: Efficient Authentication in the Ubiquitous World," The Second International Symposium on Ubiquitous Intelligence and Smart Worlds (UISW2005), Lecture Notes in Computer Science 3823, Springer-Verlag, pp.480–489, 2005

[26] Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. "Proxy Signature without Random Oracles," The Second International Conference on Mobile Ad Hoc and Sensor Networks (MSN2006), Lecture Notes in Computer Science 4325, Springer-Verlag, 2006. pp. 473–484

Jiguo Li was born in Heilongjiang Province, China, on December 17, 1970. He received his Bachelor degree from Heilongjiang University, China in 1996. He received his Master degree and PhD from Harbin Institute of Technology, China in 2000, and 2003 respectively. He is currently Associate Professor at the College of Computer and Information Engineering, Hohai University, Nanjing, China. His major interests are cryptography theory and technology, cryptography protocol and network and information security. He has published over 30 research papers.

© 2009 ACADEMY PUBLISHER