A Certificate-based Signature Scheme for Mobile Ad Hoc Networks

Zhimin Li, Xin Xu, Cunhua Li
School of Computer Engineering, Huaihai Institute of Technology, Jiangsu, China
lizhimin1981@gmail.com, xinxu@gmail.com, cli@gmail.com

Abstract

In order to insure the security of communication in mobile Ad hoc networks, we proposed a new signature scheme which is based on bilinear pairing and certificate-based cryptography. The security of the scheme was proved under the Random Oracle Model. The scheme is also efficient, since the signing algorithm does not need the computation of the bilinear pairing and the verification algorithm needs that computation only once. Thus it is particularly useful in Ad hoc networks.

Keywords: Ad Hoc Networks, Signature, Certificate-Based, Provably Secure, Bilinear Pairings

1. Introduction

Mobile Ad Hoc Networks is a new network structure which having lots of autonomous nodes that communicate with each other by forming a multi-hop wireless network. It has wide range of applications, such as industrial, commercial, medical, home, office environment etc. Especially, it can be used in military battlefield, emergency disaster and other special occasions where geographical or terrestrial constraints demand totally distributed networks. As a new type of wireless communication networks, it has the following characters, no central control, changing in the topology dynamic, distributed management, resource-constrained (limited bandwidth of wireless communication and limited host energy). So there are more obstacles compared with the traditional fixed network in information authentication, confidentiality, integrity and anti-repudiation security maintenance.

1.1. Certificate-based crypto-system

In traditional Public Key Signatures (PKS), the public key of the signer is essentially a random bit string picked from a given set. It has a problem that the authorization of the signer with the signature only cannot be satisfied. This problem can be solved by using a certificate which provides a trusted link between the public key and the identity of the signer by the CA’s signature. Moreover, Public Key Infrastructure (PKI) is designed to issue and manage certificates. In general, the signer registers its own public key with its identity in certificate server and anyone who wants to obtain the signer’s public key requests it by sending the server the identity of the signer and gets it. Before verifying a signature using the signer’s public key, the verifier must obtain the signer’s certification status which is in general by making a query on the signer’s certificate status to the CA. This course is called third-party query and it is considered as a problem in the PKI. This problem can be surmounted in signature schemes simply by transmitting the certificate for its valid public key with its signature. Despite of this settlement, a verifier must verify the certificate first and if authorization of the CA about the signer’s public key is valid then verifies the signed message with given public key from the signer. In the point of a verifier, two verification operations for independent signatures are required. This has been regarded as a drawback due to additional computation time and storage. The apparent need for this infrastructure is often cited as a reason for the widespread use of public-key cryptography.

To simplify key management procedures of conventional PKI, Shamir [1] introduced the concept of Identity-based Cryptography (IBC), in which user’s identity is as public key to reduce the requirement on the infrastructure, and he also proposed an identity-based signature scheme. The first practical provably secure Identity-based Encryption (IBE) scheme [2] was proposed by Boneh and Franklin based on bilinear map. In IBC, certification becomes implicit since the sender of a message does not need to check whether the user is certified or not. But there exists another trusted third party called Private Key Generator (PKG). PKG generates and sends user’s private key via an authentication and secure channel. The main practical benefit of IBC lies in greatly reduction of need for public key certification. However, a new problem appears in the IBC, that is, private key escrow problem as PKG.
can generate the secret keys of all its users. Moreover, secret keys generated by PKG for users must be sent over secure channel, which makes secret key distribution becomes a daunting task [3].

To solve the problems in traditional PKI and IBS, Gentry proposed the concept of Certificate-based Encryption (CBE) which combines public-key encryption (PKE) and IBE while preserving their features. In PKE, user needs to generate his own public key and secret key pair with a certificate requested from the CA. In CBE, user also generates his own secret/public key pair and requests a certificate from the CA. This certificate has all of the functionalities of a conventional PKI certificate. In addition, it can also be used as a decryption key. This additional functionality gives us an implicit certification so that the sender can doubly encrypt the message so that the recipient can decrypt it using his private key along with an up-to-date certificate from his CA. This feature allows us to eliminate the third-party queries for the certificate status. Moreover, CBE does not have key escrow problem and secret key distribution problem since the CA does not know the personal secret keys of users, and the CA’s certificate need not to be kept secret. Gentry also demonstrated how certificate-based encryption could be used to construct an efficient PKI [3] requiring less infrastructure than previous proposals.

Yum and Lee [4] revisited the definitions and security notions of certificate-based encryption and provided a formal equivalence theorem among identity-based encryption, certificate-less encryption and certificate-based encryption. Galindo [5] pointed out that a dishonest authority could break the security of the three generic constructions of CBE and CL-PKE schemes given in [6,7]. Al-Riyami and Paterson al gave an analysis of Gentry’s CBE concept, repaired a number of problems with the original definition and security model for CBE and provided a generic conversion showing that a secure CBE scheme could be constructed from any secure CL-PKE scheme. Formally, some provably secure certificate-based algorithms are designed for in Ad hoc network security communications [8-13].

1.2. Our contribution

In this paper, we proposed a certificate-based signature scheme that can be used in Ad hoc networks. The scheme was proved to be secure that it is existentially unforgeable against adaptive chosen message attacks under the computational Diffie-Hellman assumption in the random oracle. Our scheme maintains most of the advantages of CBE over PKE and IBE. Since the CA does not know user’s personal secret key, CBS does not suffer from the key escrow property which is inherent in IBC. Moreover, there is also no secret key distribution problem since the CA’s certificate need not be kept secret.

1.3. Organization of the paper

In Section 2, we review some preliminaries required in this paper and describe the security and adversarial model. In Section 3, we propose a new certificate-based signature scheme. The security proof of the scheme is provided in Section 4. We conclude the paper in Section 5.

2. Preliminaries

In this section, we review some background knowledge including the bilinear pairing and computational Diffie-Hellman (CDH) problem. We also provide the generic mode and security notions of certificate-based signature schemes.

2.1. Bilinear pairings and diffie-hellman problem

Let $G_1$ denote an additive group of prime order $p$ and $G_2$ be a multiplicative group of the same prime order. The bilinear pairing definition is usually described as follows. Let $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a bilinear mapping with the following properties.

1. Bilinear: For all $P, Q \in G_1$, and $a, b \in \mathbb{Z}_q$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$,

2. Non-degenerate: $\hat{e}(Q, R) \neq 1$, for some $Q, R \in G_1$.

3. Computable: For any $P, Q \in G_1$, there is an efficient algorithm to compute $\hat{e}(P, Q)$.

The security of our scheme relies on the hardness of the computational Diffie-Hellman problems.


Definition 1. Let \((G_1, +)\) be a cyclic additive group generated by \(P\), the computational Diffie-Hellman (CDH) problem in \(G_1\) is to compute \(abP\) given \(aP, bP\).

The success probability of any probabilistic polynomial-time algorithm \(A\) in solving the CDH problem in \(G_1\) is defined to be \(\text{Succ}^{\text{CDH}}_A = \Pr[A(P,aP,bP) = abP, a,b \in \mathbb{Z}_p]\).

The CDH assumption states that for every probabilistic polynomial-time algorithm \(A\), \(\text{Succ}^{\text{CDH}}_A\) is negligible.

2.2. Outline of certificate-based signature

A Certificate-based signature scheme consists of the following four probabilistic polynomial time (PPT) algorithms.

**Setup:** This algorithm takes as input a security parameter \(1^k\) and returns the certifier’s master secret key \(S\) and a corresponding public key \(P_{pub}\). CA also uses this algorithm to generate a public parameter \(\text{Params}\) which is shared in the system.

**KeyGen:** Given public parameter \(\text{Params}\), user uses this algorithm to produce his own secret/public key pair \((K_{ID}, P_{ID})\).

**CertGen:** When CA receives user’s ID/public key pair \((ID, P_{ID})\), he computes certificate \(\text{Cert}_{ID}\), and transmits it to its owner in a secure way.

**Sign:** To sign message \(m\), Alice obtains the ciphertext \(\sigma\) by computing \(\text{Sign}(m, K_{Alice}, \text{Cert}_{Alice})\).

**Verify:** When Bob receives the message/signature pair \((m, \sigma)\), he utilizes this algorithm with Alice’s public key \(P_{Alice}\), CA’s public key \(P_{pub}\) and system parameter to verify the \(\sigma\) validity. If \((m, \sigma)\) is valid, it outputs \textit{true}, otherwise, outputs \textit{false}.

2.3. Security notions

Roughly speaking, the security of a certificate-based signature requires that one can generate a valid signature under the public key \(P_{ID}\) if and only if he has \(\text{Cert}_{ID}\) and \(K_{ID}\). One cannot generate a valid signature only with one of those. The security of certificate-based signatures is defined by the game between the challenger \(C\) and the adversary \(A\). There are two kinds of inherent adversaries for certificate-based signature scheme, one is an adversary who knows the secret key \(K_{ID}\) but the certificate \(\text{Cert}_{ID}\) is unknown, called user forgery, the other knows the master secret key \(S\) and user’s secret key \(K_{ID}\) is unknown, called CA forgery.

Definition 2: A Certificate-based signature scheme is said to be secure against user forgery if no polynomially bounded adversary has non-negligible advantage in the following game.

**Setup:** The challenger \(C\) runs the \textit{Setup} algorithm with a security parameter \(1^k\) and obtains public parameters \(\text{Params}\) and the master private key \(S.\) \(C\) sends \(\text{Params}\) to the adversary \(A\) and keeps \(S\) secret.

**Phase I:** The adversary \(A\) performs a polynomially bounded number of queries to \(C.\) The queries made by \(A\) may be adaptive, i.e. current query may depend on the answers to the previous queries. The various oracles and the queries made to these oracles are defined below:

1. **Key queries:** \(A\) produces an identity \(ID\), \(C\) computes \(\text{KeyGen}(ID) = (K_{ID}, P_{ID})\), and sends them to \(A\).
2. **Key replacements:** \(A\) can replace user’s key arbitrarily. \(A\) produces \((ID, K_{ID}, P_{ID})\), \(C\) verifies the validity of \(PK_{ID}\) for \(K_{ID}\) correspondingly. If true, \(C\) replaces user’s previous key by \((K_{ID}, P_{ID})\), or else, sends invalid to \(A\).
3. **Certificate queries:** \(A\) produces \((ID, P_{ID})\) and sends them to \(C\), \(C\) computes \(\text{CertGen}(ID) = (K_{ID}, P_{ID}) = \text{Cert}_{ID}\), and sends \(\text{Cert}_{ID}\) to \(A\).
4. **Hash queries:** \(A\) can send query to \(C\) for the outputs of the hash function used in the sign process. \(C\) computes and sends corresponding values to \(A\).
5. **Signature queries:** \(A\) produces the sender identity ID, message \(m\), and requests the signature of \(m\). \(C\) generates the private key \(K_{ID}\) and \(\text{Cert}_{ID}\) to perform the algorithm \(\sigma = \text{Sign}(m, K_{ID}, \text{Cert}_{ID})\). \(C\) sends \(\sigma\) to \(A\).

**Phase II ( Forgery):** After a sufficient amount of training, \(A\) produces a signature \((m, \sigma, ID)\) to \(C.\) Here, \(A\) should not have queried the certificate of ID and \(\text{Sign}(m, ID)\) during phase I. \(A\) wins the game, if \(\text{Unsign}(m, \sigma, ID)\) is valid.

The advantage \(\text{Adv}(A)\) of \(A\) is defined as the probability that it wins.
Definition 3: A Certificate-based signature scheme is said to be secure against CA forgery if no polynomially bounded adversary has non-negligible advantage in the following game.

**Setup:** The challenger $C$ runs the Setup algorithm with a security parameter $1^k$ and obtains public parameters $\text{Params}$ and the master private key $S$. $C$ sends $\text{Params}$ and another private key $S$ to the adversary $A$.

**Phase I:** The adversary $A$ performs a polynomially bounded number of queries to $C$. The queries made by $A$ may be adaptive, i.e. current query may depend on the answers to the previous queries. The various oracles and the queries made to these oracles are defined below:

1. **Public key queries**: $A$ produces an identity $\text{ID}_i$, $C$ computes $\text{KeyGen}(\text{ID}) = (K_{\text{ID}}, PK_{\text{ID}})$, and sends $PK_{\text{ID}}$ to $A$.
2. **Private key queries**: $A$ produces $(\text{ID}, PK_{\text{ID}})$ to $C$, $C$ verifies whether $PK_{\text{ID}}$ exists or not. If true, $C$ sends $K_{\text{ID}}$ correspondingly to $A$, or else, sends invalid to $A$.
3. **Certificate queries**: $A$ produces $(\text{ID}, PK_{\text{ID}})$ and sends them to $C$, $C$ computes $\text{CertGen}(\text{ID}) = (K_{\text{ID}}, PK_{\text{ID}}) = \text{Cert}_{\text{ID}}$, and sends $\text{Cert}_{\text{ID}}$ to $A$.
4. **Hash queries**: $A$ can send query to $C$ for the outputs of the hash function used in the sign process.

$C$ computes and sends corresponding values to $A$.
5. **Signature queries**: $A$ generates the sender identity $\text{ID}$, and requests the signature of $m$. $C$ computes the private key $K_{\text{ID}}$ and $\text{Cert}_{\text{ID}}$ to perform the algorithm $\sigma = \text{Sign}(m, K_{\text{ID}}, \text{Cert}_{\text{ID}})$. $C$ sends $\sigma$ to $A$.

**Phase II (Forgery):** After a sufficient amount of training, $A$ produces a signature $(m, \sigma, \text{ID})$ to $C$. Here, $A$ should not have queried the certificate of $\text{ID}$ and $\text{Sign}(m, \text{ID})$ during phase I. $A$ wins the game, if $\text{Unsign}(m, \sigma, \text{ID})$ is valid.

The advantage $\text{Adv}(A)$ of $A$ is defined as the probability that it wins.

In definition 2, the adversary does not know the master private key, but he can replace user’s private/public key pair arbitrarily. So if the scheme is secure against user forgery, the scheme is also secure against key replacement attack.

**3. Concrete scheme**

In this section, we propose a new Certificate-based signature scheme which can be efficient used in Ad hoc Networks. The following shows the details of our scheme.

**Setup:** Define $G_1, G_2$ and $\hat{e}$ as in previous section. Let $H_1, H_2$ and $H_3$ be three cryptographic hash functions where $H_1: \{0, 1\}^* \rightarrow G_1$, $H_2: \{0, 1\}^* \rightarrow G_1$, $H_3: \{0, 1\}^* \rightarrow G_1$. Let $P$ be a generator of $G_1$, CA chooses a master secret key $S \in \mathbb{Z}_q^*$, keeps $S$ secret and computes $P_{\text{pub}} = SP$. The system’s public parameters $\text{Params}$ are $(G_1, G_2, q, P, P_{\text{pub}}, \hat{e}, H_1, H_2, H_3)$.

**Key Extract:** For given $\text{Params}$, user with ID chooses $x \in \mathbb{Z}_q^*$ as his private key and generate his public key by computing $PK_{\text{ID}} = xP$.

**Certificate Extract:** To generate a certificate for a user with $(\text{ID}, PK_{\text{ID}})$, $C$ computes $C_\text{ID} = SH_1(\text{ID}, PK_{\text{ID}})$ as user’s certificate.

**Sign:** To sign a message $m \in \{0, 1\}^*$, user $A$ with identity $\text{ID}_A$ follows the steps below.

1. Choose $r \in \mathbb{Z}_k$
2. Compute $U = rP$
3. Compute $W_1 = H_3(m, \text{ID}_A, PK_A)$.
4. Compute $W_2 = H_3(m, U \oplus W_1)$.
5. Compute $V = C_\text{ID} + K_{\text{ID}}W_1 + rW_2$.

The signature is $\sigma = (U, V)$.

**Verify:** Given a message/signature pair $(m, U, V)$, the system parameter $\text{params}$ and user A’s public key $PK_A$, user B works as follows:

1. Compute $Q = H_1(\text{ID}_B, PK_B)$.
2. Compute $W'_1 = H_3(m, \text{ID}_B, PK_B)$.
3. Compute $W'_2 = H_3(m, U \oplus W'_1)$.
4. Check the equation $\hat{e}(V, P) = \hat{e}(P_{\text{pub}}, Q)\hat{e}(W'_1, PK_A)\hat{e}(W'_2, U)$. If the equality holds, outputs $\text{true}$. Otherwise, reject.
4. Security analysis

In this section, we analyze the security of the proposed scheme by using the definitions mentioned in Section II. The analysis results show that the scheme is secure against both user and CA forgery.

**Theorem 1.** If there exists an adversary called A that is able to make a user forgery with an advantage \( \epsilon \), then there exists a distinguisher C that can solve the CDH problem with advantage \( O(\epsilon) \).

**Proof.** The interaction between A and C can be viewed as a game given in definition 2. When C is provided with a random instance \((P, aP, bP)\) of the CDH problem, C can use A as a subroutine and act as A’s challenger in the user forgery game to compute \( abP \). In the proof, the hash functions are regarded as the random oracles. During the game, A will consult C for answers to the random oracles \( H_1, H_2 \) and \( H_3 \). C maintains lists \( L_1, L_2, L_3 \) respectively in giving the responses to the queries. These answers are randomly generated, but to maintain the consistency and to avoid collision.

**Setup:** For having the game with \( A, C \) choose \( P_{pub} = aP \) and gives \( A \) the system parameters \((G_1, G_2, q, P, P_{pub})\). Note that \( a \) is unknown to \( C \), this value simulates the master secret key value for \( \text{CA} \) in the game.

**Phase I:** During phase I, \( A \) is allowed to access the various oracles provided by \( C \). \( A \) can get sufficient training before generating the forgery. The various oracles provided by \( C \) to \( A \) during training are as follows.

- **\( H_1 \) Oracle Queries (\( \Omega_{\Omega_1} \)):** When \( A \) makes a query on \( H_1 \), \( C \) performs as follows. If \((ID, PK_i) \) is available in list \( L_1, C \) returns the corresponding value to \( A \). Otherwise, on a new \( H_1 \) query \((ID, PK_i)\), \( C \) responds as follows. \( C \) first chooses a random number \( t \in \{0, 1\} \) such that \( Pr[t = 1] = \delta \) where the value of \( \delta \) will be determined later.
  
  \( 1 \) If \( t = 1 \), \( C \) chooses a random number \( a_t \in \mathcal{I} \) and sets \( \Omega_{\Omega_1}(ID, PK) = a_tP + bP \) where \( bP \) is another random value of CDH problem and stores \((ID, PK, a_t, bP)\) in list \( L_1 \). \( C \) returns \( a_tP + bP \) to \( A \).
  
  \( 2 \) Else \( t = 0 \), \( C \) chooses a random number \( a_t, a_t \in \mathcal{I} \), and sets \( \Omega_{\Omega_1}(ID, PK) = a_tP \) and stores \((ID, PK, a_t, a_tP)\) in list \( L_1 \). \( C \) returns \( a_tP \) to \( A \).

- **\( H_2 \) Oracle Queries (\( \Omega_{\Omega_2} \)):** When \( A \) makes a query with input \((m, ID, PK_i)\), \( C \) performs the following. If \((m, ID, PK, \beta, \beta P)\) is available in list \( L_2 \), \( C \) returns \( \beta P \) to \( A \). Otherwise, \( C \) picks \( \beta \in \mathcal{I} \) satisfying no vector \((\cdot, \cdot, \cdot, \beta, \beta P)\) exists in \( L_2 \). Then, \( C \) chooses \( a_t \) randomly, and sets \( \Omega_{\Omega_2}(m, ID, PK) = a_tP \) stores \((m, ID, PK, a_t, a_tP)\) in list \( L_2 \). Then, \( C \) returns \( a_tP \) to \( A \).

- **\( H_3 \) Oracle Queries (\( \Omega_{\Omega_3} \)):** On a \((m, U)\) query, \( C \) checks whether there exists \((m, U, \gamma, \gamma P)\) in \( L_3 \) or not. If such a tuple is found, \( C \) answers \( \gamma \), otherwise \( C \) chooses \( \gamma \in \mathcal{I} \) and sets \( \Omega_{\Omega_3}(m, U) = \gamma P \) returns it as an answer to \( A \) and puts the tuple \((m, U, \gamma, \gamma P)\) into \( L_3 \).

- **Key extraction queries:** When \( A \) asks the secret/public key pair of user with identity \( ID_i \). \( C \) firstly checks whether there exists \((ID_i, \cdot, \cdot)\) in the key-list \( L_4 \) or not. If exists, \( C \) returns the corresponding public key \( PK_i \) to \( A \). Otherwise \( C \) chooses \( s \) randomly, and sets \((K_i, PK) = (s, sP)\). Then, \( C \) adds \((ID_i, K_i, PK)\) into the list \( L_4 \) and returns \( PK_i \) to \( A \).

- **Key replacements:** \( A \) submits \((ID, K_{ID}, PK_{ID})\) to \( C \). \( C \) verifies the validity of \( PK_{ID} \) for \( K_{ID} \). If true, \( C \) replaces user’s previous key by \((K_{ID}, PK_{ID}) \) in \( L_2 \), or else, returns invalid to \( A \).

- **Certificate queries:** On a certificate query \( ID_i \) \( C \) first checks the certificate-list \( L_4 \) whether there exists \((ID_i, \cdot, \cdot)\) or not. If exists, \( C \) returns the corresponding certificate \( C_i \) to \( A \). Otherwise, \( C \) checks \( L_1 \) to obtain this user’s public key \( PK_i \). Here, we assume that \((ID_i, PK_i, \cdot, \cdot)\) has been in list \( L_1 \). Otherwise, \( C \) can add \((ID_i, PK_i, \cdot, \cdot)\) into \( L_1 \) as the same way that he responds to \( H_1 \) oracle queries. If the output of \( H_1 \) oracle queries is \( a_tP \), \( C \) stores the corresponding certificate \( C_i = a_tP_{pub} \) in \( L_1 \) and returns it to \( A \). Otherwise, \( C \) aborts.

- **Signature queries:** On a sign query of message \( m \) with identity \( ID_i \), \( C \) first checks list \( L_1 \) to obtain the output value of \( H_1 \). If the responding value is \( a_tP \), \( C \) can generate the certificate \( C_i \) as he responds the certificate queries and use \((C_i, K_i)\) to sign the message \( m \). Otherwise, \( C \) chooses \( r_i \in \mathbb{Z}_q^* \), and sets \( U_i = r_iP - P_{pub} \). Then \( C \) does the follows.

1. Firstly, \( C \) checks \( L_2 \) whether \((m, ID, PK, \cdot, \cdot)\) exists or not. If that is not true, \( C \) adds \((m, ID, PK, \beta, \beta P)\) into \( L_2 \) as the same way he responds to \( H_1 \) oracle queries.
2. Secondly, \( C \) checks whether \((m, U, \cdot, \cdot)\) exists in \( L_3 \). If it does, \( C \) must rechoose the number \( r_i \) until there is no collision. \( C \) further sets \( \Omega_{\Omega_3}(m, U) = \gamma P + bP \) and adds \((m, U, \gamma, \gamma P + bP)\) into list \( L_3 \).
3. At last, $C$ computes $V_i = aP_{pub} + K_i\Omega_D(m_i, ID_i, PK_i) + \gamma_iU_i + r_iP$ and outputs $(U_i, V_i)$ as the signature.

**Phase II (Forgery):** After getting sufficient queries, $A$ submits the signature $(m*, \sigma* = (U*, V*))$, $ID_i$, $PK_i$, $C_i$ with the following restrictions that $A$ has not ever queried the certificate of $ID_i$ and the corresponding signature of $m*$. We assume that $(ID_i, PK_i)$, $(m*, ID_i, PK_i, \beta*, \beta*P)$, $(m*, \gamma*, \gamma*P)$ have been in the lists $L_1$, $L_2$ and $L_3$ respectively. If such a tuple is found, $V^* = a\Omega_D(ID_i, PK_i) + \beta*PK_i + \gamma*P^*$. If $t = 1$, $\Omega_D(ID_i, PK_i) = aP + bP$. Therefore, $C$ can compute $abP = V^* - (aP_{pub} + \beta*PK_i + \gamma*P^*)$. Otherwise, $C$ fails to solve this CDH problem instance.

**Probability Analysis:** According to the simulation, $C$ can derive the value of $abP$ if and only if all the following three events happen. Firstly, $C$ does not fail during the simulation, note this event as $E_1$. Secondly, $A$ outputs a valid forgery, note it as $E_2$. At last, $t = 1$ in the forgery output by $A$, note it as $E_3$. Therefore, the probability that $C$ can solve this instance of CDH problem is computed as follows.

$$\text{Succ}_{\text{CDH}} = \Pr[E_1 \land E_2 \land E_3] = \Pr[E_1] \Pr[E_2|E_1] \Pr[E_3|E_1 \land E_2].$$

In addition, all the simulation can be done in polynomial time. From the simulation, we have $\Pr[E_1] \geq (1 - \delta)^q$, $\Pr[E_2|E_1] = \text{Adv}(A) = \epsilon$ and $\Pr[E_3|E_1 \land E_2] = \delta$. Thus, we have the conclusion that

$$\text{Succ}_{\text{CDH}} \geq \delta(1 - \delta)^q \epsilon.$$ 

Thus, if the advantage $\epsilon$ of $A$ to give a user forgery of the scheme is non-negligible, the probability of $C$ solving CDH problem is also non-negligible. \qed

**Theorem 2.** If their exists an adversary called $A$ that is able to make a CA forgery with an advantage $\epsilon$, then there exists a distinguisher $C$ that can solve the CDH problem with advantage $O(\epsilon)$.

**Proof.** The interaction between $A$ and $C$ can be viewed as a game given in definition 3. Like Theorem 1, when $C$ is provided with a random instance $(P, aP, bP)$ of the CDH problem, $C$ uses $A$ as a subroutine and acts as $A$’s challenger in the CA forgery game to compute $abP$. In the proof, the hash functions are regarded as the random oracles. During the game, $A$ will consult $C$ for answers to the random oracles $H_1, H_2$ and $H_3$. $C$ maintains lists $L_1$, $L_2$, $L_3$, $L_4$ respectively in giving the responses to the queries of these oracles as well as key and certificate queries. These answers are randomly generated, but to maintain the consistency and to avoid collision.

**Setup:** For having the game with $A$, $C$ chooses a random number $S \in \mathbb{Z}_*$ as the master secret key, sets $P_{pub} = SP$ and gives $A$ the system parameters $(G_1, G_2, q, P, S, P_{pub})$.

**Phase I:** During phase I, $A$ is allowed to access the various oracles provided by $C$. $A$ can get sufficient training before generating the forgery. The various oracles provided by $C$ to $A$ during training are as follows.

- **$H_1$ Oracle Queries ($\Omega_{H_1}$):** When $A$ makes a query on $H_1$, $C$ performs as follows. If $(ID_i, PK_i, )$ is available in list $L_1$, $C$ returns the corresponding value to $A$. Otherwise, On a new $H_1$ query $(ID_i, PK_i)$, $C$ chooses a random number $\alpha_i \in \mathbb{Z}_*$, sets $\Omega_{H_1}(ID_i, PK_i) = \alpha_iP$ and stores $(ID_i, PK_i, \alpha_i, \alpha_iP)$ in list $L_1$. $C$ returns $\alpha_iP$ to $A$.

- **$H_2$ Oracle Queries ($\Omega_{H_2}$):** When $A$ makes a query with input $(m, ID_i, PK_i)$, $C$ performs the following. If $(m, ID_i, PK_i)$ is available in list $L_2$, $C$ returns the corresponding value to $A$. Otherwise, $C$ picks $\beta \in \mathbb{Z}_*$ satisfying no vector $(\cdot, \cdot, \cdot, \cdot, \cdot) \in \mathbb{Z}_*$, sets $\Omega_{H_2}(m, ID_i, PK_i) = \beta P + bP$ stores $(m, ID_i, PK_i, \beta, \beta + bP)$ in list $L_2$, where $bP$ is the random value of CDH problem Then, $C$ returns $\beta P + bP$ to $A$.

- **$H_3$ Oracle Queries ($\Omega_{H_3}$):** On a $(m, U_i)$ query, $C$ checks whether there exists $(m_i, U_i, \gamma_i, \gamma_iP)$ in $L_3$ or not. If such a tuple is found, $C$ answers $\gamma_i$, otherwise he chooses $\gamma \in \mathbb{Z}_*$, sets $\Omega_{H_3}(m, U_i) = \gamma P$ returns it as an answer to $A$ and puts the tuple $(m_i, U_i, \gamma, \gamma P)$ into $L_3$.

- **Key extraction queries:** When $A$ asks the public key pair of user with identity $ID$, $C$ firstly checks whether there exists $(ID, \cdot, \cdot)$ in the key-list $L_4$ or not. If exists, $C$ returns the corresponding public key $PK_i$ to $A$. Suppose there are up to $q_k$ key queries, $C$ will choose a random number $t \in \{1, 2, \ldots, q_k\}$. If $ID$ is the $t^{th}$ query, $C$ sets $K_i = \bot$ and $PK_i = aP$ where $aP$ is the input random value of CDH problem. Here, the symbol $\bot$ means $B$ doesn’t know the corresponding value. Otherwise, $C$ chooses $s_i$ randomly, and sets $(K_i, PK_i) = (s_i, s_iP)$. In both cases, $C$ adds $(ID_i, K_i, PK_i)$ into the list $L_4$ and returns $PK_i$ to $A$.

- **Signature queries:** After a sign query of message $m$ with identity $ID_i$, $C$ first checks list $L_4$ to obtain the corresponding secret key $K$. If $K_i = \bot$, $C$ chooses two random elements $U_i = rP \in G_1$, and $b_i \in \mathbb{Z}_*$, $rP$. Then, he adds $(m, U_i, PK_i, \beta, \beta P)$ into List $L_2$. If a collision occurs, $U_i$ and $b_i$ should be re-chosen. In addition, $C$ adds $(m, U_i, PK_i, \beta, \beta P)$ to List $L_3$ as the same way he responds to $H_3$ queries. By assumption,
(ID, PKi, α, αP) has been in List L1, C computes V1 = Cert1 + βPKi + γU1. Otherwise, C can use Kα and Cert1 to generate the signature on the message. Then, C returns signature (Uα, V1) to A.

**Corruption:** On a corruption query IDα, C will check the list Lα and return Kα to A. If Kα = ⊥, C fails to solve this CDH problem.

**Phase II (Forgery):** After getting sufficient queries, A submits the signature (m*, α* = (U*, V*), IDα, PKi, C) with the following restrictions that A has not ever queried the private key of IDα and the corresponding signature of m*. We assume that (IDα, PKi, α), (m*, α), PKi, β*, βP), (m*, U*, γ*, γP) have been in the lists L1, L2 and L3 respectively. If (U*, V*) is a valid signature of the message m*, then V* = S(tαP + KαβP + βPγ) + γ*U*. If i = t, i.e., Kα = a, PKi = αP. Therefore, C can compute abP = V* − (S(tαP + βPγ) + γ*U*). Otherwise, C fails to solve this CDH problem instance.

**Probability Analysis:** According to the simulation, C can derive the value of abP if and only if all the following three events happen. Firstly, C does not fail during the simulation, note this event as E1. Secondly, A outputs a valid forgery, note it as E2. At last, i = t in the forgery output by A, note it as E3. Therefore, the probability that C can solve this instance of CDH problem is computed as follows. \( \text{Succ}_C^{\text{CDH}} = \Pr[E_1 \land E_2 \land E_3] = \Pr[E_1] \Pr[E_2|E_1] \Pr[E_3|E_1 \land E_2] \).

In addition, all the simulation can be done in polynomial time. Assume \( q_{\Omega_1}, q_{\Omega_2}, q_{\Omega_3}, q_{\Omega_4}, q_{\Omega_5}, q_{\Omega_6} \) are the maximum polynomial number of queries allowed to the oracles \( \Omega_1, \Omega_2, \Omega_3, \Omega_4, \Omega_5, \Omega_6 \), key extraction, sign queries respectively. From the simulation, we have \( \Pr[E_1] \geq (1 - \frac{1}{q_{\Omega_6}})^{|S|} \), \( \Pr[E_2|E_1] = \text{Adv}(A) = e \) and \( \Pr[E_3|E_1 \land E_2] = 1/q_{\Omega_8} \). Thus, we have the conclusion that \( \text{Succ}_C^{\text{CDH}} \geq \frac{1}{q_{\Omega_6}} (1 - \frac{1}{q_{\Omega_6}})^{|S|} \).

Thus, if the advantage \( e \) of A to give a user forgery of the scheme is non-negligible, the probability of C solving CDH problem is also non-negligible. □

### 5. Efficiency comparison

In this section, we make a comparison between our proposed scheme and other certificate-based signature schemes which are also provably secure under random oracle. The following notations will be used in the comparison: M scalar quantity multiplication like \( aP \) in \( G_1/G_2 \), SM scalar quantity multiplication like \( aP + bQ \) in \( G_1/G_2 \), E exponentiation in \( G_1 \), P bilinear pairing operation. The following table shows the comparison of our scheme and the schemes proposed in [10, 11]. As shown in the table, our scheme enjoys shorter signature length under the same system parameters. Meanwhile, our scheme also requires less operation cost under the assumption that the pairing \( e(P_{pub}, Q) \) in verification algorithm can be pre-computed.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Sign</th>
<th>Verify</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheme in [10]</td>
<td>3M</td>
<td>3P</td>
<td>(</td>
</tr>
<tr>
<td>Scheme in [11]</td>
<td>1M + 2SM</td>
<td>3P</td>
<td>(</td>
</tr>
<tr>
<td>Our Scheme</td>
<td>3M</td>
<td>1M + 2SM + 2P</td>
<td>(</td>
</tr>
</tbody>
</table>

### 6. Conclusion

In this paper, we have proposed a provably secure certificate-based signature scheme that can be used in Ad hoc networks. Under the assumption that the well-known computational Diffie-Hellman problem is difficult, we have formally proved the security of the newly proposed scheme in the random oracle model, that our proposal is proven secure against both user forgery and CA forgery. Compared with the other secure certificate-based signature schemes [10, 11], our scheme enjoys shorter signature length and less operation cost. Therefore, our scheme is suitable for use in Ad hoc networks.
7. Acknowledgment

This work is sponsored by Jiangsu Provincial Natural Science Foundation of China (No.BK20082140), Item of Scientific Research Fund for Talents of Huaihai Institute of Technology (No.KQ10121), Research Fund of Huaihai Institute of Technology (No.KX10530).

8. References