A Certificate-Based Proxy Signature with Message Recovery without Bilinear Pairing

Ali Mahmoodi[1],Javad Mohajeri[2], Mahmoud Salmasizadeh[2]
Electrical Engineering Dept.[1], Electronics Research Institute[2]
Sharif University of Technology
Tehran, Iran
mahmoodi@ee.sharif.edu[1],{mohajer,salmasi}@sharif.edu[2]

Abstract.

In this paper, we propose the first provable secure certificate-based proxy signature with message recovery without bilinear pairing. The notion of certificate-based cryptography was initially introduced by Gentry in 2003, in order to simplify certificate management in traditional public key cryptography (PKC) and to solve the key escrow problem in identity-based cryptosystems. To date, a number of certificate-based proxy signature (CBPS) schemes from bilinear pairing have been proposed. Nonetheless, the total computation cost of a pairing is higher than that of scalar multiplication (e.g., over elliptic curve group). Consequently, schemes without pairings would be more appealing in terms of efficiency. According to the available research in this regard, our scheme is the first provable secure CBPS scheme with message recovery which is based on the elliptic curve discrete logarithm problem. We prove the security of the presented scheme against existential forgery under adaptive chosen message and ID attacks in the random oracle model. Moreover, the paper will also show how it would be possible to convert this scheme to the CBPS scheme without message recovery. This scheme has more applications in situations with limited bandwidth and power-constrained devices.

Keywords

Proxy signature; certificate-based; message recovery; bilinear pairing; ECDLP

1. Introduction

Proxy delegation of legal affairs is known as a common practice in formal interactions. In order to simplify the representation of the proxy signatures in the electronic world, the concept of proxy signature was first introduced by Mambo et al. in 1996 [1] which allows a designated person, called proxy signer, to sign on behalf of the original signer. The proxy signature plays an important role in many applications such as wireless e-commerce, mobile agents, and etc. [2]. However, the theory of proxy signature confronts with some problems when it comes to reality. In traditional public key cryptosystems (TPKC) or Public Key Infrastructure (PKI) [3], the communication and the validation (third party query, etc.) of a large number of public key of users, greatly affect the efficiency of the proxy signature schemes [4]. To simplify certificate management and to improve the proficiency of traditional PKI, Shamir [5] introduced Identity-based cryptology (IBC). It

can solve the above-mentioned problem by using signer's identity such as email address or IP number as his public key while the corresponding private key will be a result of some mathematical operation that takes as input the user's identity and the master secret key of a trusted authority, referred to as Private Key Generator (PKG). The main practical advantage of IBC is in greatly reducing the need for public key certification. Certificate is provided implicitly in IBC and an explicit authentication of public key isn't required. The PKG generates the private key of all its users; therefore, key escrow becomes an inherent problem in IBC. It results in unconditional trust to PKG for a user. In addition, the PKG must send produced private key over secure channel, which makes secret key distribution a daunting task [6].

To fill the gap between traditional PKC and IBC, Al-Riyami and Paterson proposed a new paradigm called Certificateless public key cryptography (CLPKC) [7]. In CLPKC, the KGC generates a partial secret key from the user's identity using the master secret key, while the user independently generates his own private/public keys. Decryption and signature generation require both the user private key and partial secret key. Therefore, CLPKC not only solves the key escrow problem, but also eliminates the use of certificates as in TPKC. Due to the lack of public key certificate, it is significant to assume that an adversary in the CLPKC system can replace the user's public key with any false key of its choice, which is also known as key replacement attack [4]. Cryptographic protocols in CLPKC system are easily suffered from this kind of attack. For example, the first CL-based signature scheme [7] is not secure against the key replacement attack [8, 9]. This problem was later fixed. We will not go through the detail, since it is out of the scope of the present paper. To read about the details of the key replacement attack in CL system the interested readers can refer [8, 9].

In order to use the advantageous of the identity-based cryptology in the traditional cryptology, Gentry [6] introduced the notion of certificate-based encryption (CBE) in 2003, which combines public-key encryption (PKE) with IBE by preserving their merits. In CBE, each user generates his own private/public pair keys and requests a certificate of his public key from the CA, while the CA uses the key generation algorithm of an identity based encryption (IBE) scheme to generate the certificate. This certificate has all of the functions of a conventional PKI certificate. In addition, it can be used as a part of the signing key. The feature of implicit certification eliminates third-party queries for the certificate status. Since the CA does not know the personal secret keys of users, there is no key escrow problem in CBE, and since the CA's certificate need not be kept secret, there is no distribution of secret key problem. Nevertheless, a CBE scheme is inefficient when a CA covers a large number of users and performs frequent certificate updates [6,10,11]. However Gentry proposes using subset covers to overcome inefficiency [6]. In addition to CBE, in 2004, the notion of certificate based signature (CBS) was first proposed by Kang et.al [12]. They proposed two schemes, but unfortunately, one of their schemes was found insecure against key replacement attack [3] which was pointed out by Li et.al [13], in addition, they proposed a notion of certificate-based proxy signature model. Since 2004, many certificate-based signature schemes were given [10-14] but just a few certificate-

based proxy signature schemes have been proposed. In 2007, Wang et.al [14] proposed a certificate-based proxy cryptosystem with revocable proxy decryption power. In 2009, Jiguo Li et.al [11] proved that the certificate-based proxy signature scheme presented by Kang et.al isn't secure against key replacement attacks and they further proposed two provable secure certificate-based proxy signature schemes in the random oracle model. These two schemes don't have trust third party (CA) and in fact original signer acts as CA in the scheme. In 2010, Chen and Huang suggested a certificate-based proxy signature with trust third party but they couldn't present security proof in formal model for their scheme. Besides, Liang et.al [15] proposed a provable secure certificate-based strong designated verifier proxy signature scheme in random oracle model. All previous certificate-based proxy schemes that mentioned above require pairing operations which is one of the costly operations compared to the other operations such as Elliptic Curve exponentiation. Till now, there is not any certificate-based proxy signature scheme without bilinear pairing, that is why the present study will propose it. Accordingly schemes without pairing would be more appropriate to be implemented in power-constrained devices, such as wireless sensor networks, etc. [16].

The proxy signature scheme with message recovery is a digital signature scheme in which the original message of the signature does not need to be transmitted in addition to the proxy signature since it has been appended to the signature and can be recovered according to the verification/message recovery process. This type of signature is different from a signcryption scheme or authenticated encryption schemes, because in proxy signature with message recovery scheme, the embedded message can be recovered by everyone without the secret information. The purpose of proxy signature scheme with message recovery is to degrade the total length of the original message and the appended signature [16]. Therefore, these are useful in any application where bandwidth is one of the main concerns or small message should be signed. Recently, Singh and Verma [16] proposed the first pairing based proxy signature scheme with message recovery and proved that their scheme is provable secure but Tian et.al [17] showed that Singh's scheme isn't secure. Afterwards, Padhay et.al [4] proposed a certificateless proxy signature with message recovery and they could show that their scheme is secure against existential forgery under adaptive chosen message and ID attacks.

Our Contributions

In the present paper, we propose the first provable secure certificate-based proxy signature without pairing with message recovery scheme. In our scheme, message can be recovered from the signature; hence, message doesn't need to be transmitted along with the signature which gains smaller communication cost compared with the schemes without message recovery [4]. It is claimed by the researchers of the present study that the scheme proposed here is secure against adaptive chosen message and ID attack in random oracle model based on ECDLP. What follows represents different sections of the paper:

Section 2 defines security model of our CBPS scheme. Section 3, presents a novel CBPS scheme with message recovery scheme based on the given model and it will be shown how it would be possible to convert this scheme to the CBPS scheme without message

recovery. Then, in Section 4, the security of our scheme is analyzed. Eventually, in Section 5, concludes the paper.

2. Models of certificate-based proxy signature

2.1 Definition

Let A be an original signer with identity ID_A and B is be one of his proxy signers with identity ID_B. A warrant message m_w is used in order to delegate the signing rights which contain the identities of original and proxy signers, period of delegation, message type, scope of authorization of proxy signer, and etc. Our model is inspired by [10]. A CBPS scheme consists of four entities: certification authority (CA), original signer, proxy signer, verifier and six algorithms: Setup, UserKeyGen, CertGen, ProxyKeyGen, PSign and PSVerif/MRecovery.

1. **Setup:** This algorithm takes the security parameter δ as input and returns the system parameters $params$ and master private key s_C as output.
2. **UserKeyGen:** This algorithm takes $params$ and user identity ID_V, as input and returns user private/public pair key (PK_V, SK_V), as output.
3. **CertGen:** This algorithm takes $params$, master private key s_C, user identity ID_V and user public key PK_V as input and returns certificate $Cert_V$ as output.
4. **ProxyKeyGen:** This algorithm takes $params$, original signer public key PK_A, delegation W, (i.e., the signed warrant m_w by original signer), proxy signer private/public key pair, and public key's certificate of proxy signer $Cert_B$ as input and returns proxy signature key D_{ps}, as output.
5. **PSign:** This algorithm takes D_{ps} and message m as input and returns proxy signature S on the message m with delegation W as output.
6. **PSVerif/MRecovery:** This algorithm takes $params$, original signer's and proxy signer's identities, public keys of proxy signer and original signer and proxy signature S as input, and recovers the message m with redundant data. If the message m conforms to the redundancy data, the correctness of the proxy signature is verified.

2.2 Security Model.

To evaluate the security of CBPS schemes, according to [4,11,12], two kinds of adversaries with the different capabilities are defined: the adversary type 1, A_1, and type 2, A_2.

Adversary type 1, A_1: Acts as a malicious user which doesn't have the master private key, s_C, but can corrupt any user (including the various proxy and original signers) except target original/proxy signers. Moreover, he can replace user's public key with any value which is

chosen by him with limitation that he cannot obtain the certificate of the false public keys [11].

Adversary type 2, A_2: Acts as a malicious CA which has the master private key, s_C, but is not able to replace the user's public key [11].

Existential unforgeability against Adversary A_1

In this subsection, capabilities of A_1will be considered. Informally, the scenarios of an attack will be considered where the goal of an adversary is forging a valid CBPS under the warrant message,m_w^*, and the public key PK_V^* while, he doesn't know the certificate and the delegation of the warrant. It should be noted that the public key PK_V^* might be the fake one chosen by the adversary or the genuine one generated by the user ID. Actually, A_I has the following capabilities [11]:

1- A_1 can access some message/signature pairs (m_i, S_i) which are generated by the proxy signer with identity ID_i.
2- A_1 can replace the public key of the user with identity ID_V with PK_V^* which is chosen by himself. He can also dupe any other third party to verify user ID_V signatures with the false public key PK_V^*.
3- If A_1 has replaced the user ID_V public key with any value which is chosen by himself, he can't access the certificate of the false public key and the delegation of warrant m_w^* from the original signer.

The security of CBPS by the following game between A_1 and the challenger C is defined:

- **Setup**: The challenger C executes the **Setup** algorithm and returns the system parameters $params$ to A_1.
- **UserKeyGen Query**: On the query ID_V if ID_V already has been created, nothing is to be carried out by C. Otherwise, C runs **UserKeyGen** algorithm and access the user secret/public key pair (PK_V, SK_V), then it adds (ID_V, SK_V, PK_V) to the L_K list. In both cases, PK_V is returned to A_1.
- **Private Key Query**: On the query ID_V, C searches ID_V in the L_K list and returns private key SK_V to A_1.
- **ReplaceKey Query**: On the query ID_V and secret/public key pair (PK_V, SK_V), C searches ID_V in the L_K list. If it is not found, nothing will be carried out. Otherwise, C updates (ID_V, SK_V, PK_V) to (ID_V, SK_V^*, PK_V^*).
- **CertGen Query**: On the query ID_V and public key PK_V, the challenger C execute the **CertGen** algorithm and returns the certificate $Cert_V$ to A_1.
- **ProxyKeyGen Query**: On the query identities ID_U,ID_V the challenger C execute the **ProxyKeyGen** algorithm and returns the valid proxy key D_{ps} to A_1.
- **PSign Query**: On input of a message m, identities ID_U,ID_V and warrant m_w, the challenger C first execute the **ProxyKeyGen** and **CertGen** algorithms to access the proxy singing key, then executes the **PSign** algorithm and generates the valid proxy signature S and returns it to A_1.

- **Forge**: Finally, A_1 outputs a signature S^* with the warrant m_w^*, ID_V (or ID_V^*) and the message m^* such that:
    1- S^* is a valid CBPS on the message m^* under the public key PK_V^* and the warrant message m_w^*.
    2- ID_V^* or ID_V and PK_V^*, has not been requested as one of the **CertGen** queries.
    3- m_w^*, ID_V^* (or ID_V) and ID_U, has not been requested as one of the **ProxyKeyGen** queries.
    4- The tuple $(m^*, m_w^*, ID_V^* (or\ ID_V)\ , ID_U)$ has not been requested as one of the **PSign** queries.

In this model, the A_1 is allowed to replace the target user's public key with any value chosen by himself however, he can't obtain the certificate of the user's public key and the delegation of the warrant. Moreover, A_1 is allowed to corrupt any proxy signer except target proxy signer in the system.

Definition 1

An adversary A_1 is called a $(\epsilon, t, q_c, q_s, q_H)$-forger if it has the advantage of at least ϵ in the above game which runs in time at most t and makes at most q_c, q_s and q_H create, signing and random oracle queries, respectively. Accordingly, a scheme is called to $(\epsilon, t, q_c, q_s, q_H)$-secure against A_1 or actually unforgeable against chosen message an ID attack if no $(\epsilon, t, q_c, q_s, q_H)$-forger exists [4].

Existential unforgeability against Adversary A_2

The existential unforgeability of a CBPS for an adversaryA_2, needs that it is hardship for CA to generate a valid proxy signature of the message m^* without helping of the proxy signer. A_2 has the following capabilities [11]:

1- A_2 knows the master private key, s_C.
2- A_2 can access some message/signature pairs (m_i, S_i) which are generated by the proxy signer with identity ID_i.
3- A_2 can't replace the public key of the any user in the system.

The security of our CBPS is defined by the following game between A_2 and the challenger C.

- **Setup**: The challenger C executes the **Setup** algorithm and returns the system parameters params to A_2.
- **UserKeyGen Query**: On the query ID_V if ID_V already has been created, nothing is to be carried out by C. Otherwise, C runs **UserKeyGen** algorithm and access the user secret/public key pair (PK_V, SK_V), then it adds (ID_V, SK_V, PK_V) to the L_K list. In both cases, PK_V is returned to A_2.

- **Private Key Query**: On the query ID_V, C searches ID_V in the L_K list and returns private key SK_V to A_2.
- **ProxyKeyGen Query**: On the query identities ID_U,ID_V the challenger C executes the **ProxyKeyGen** algorithm and returns the valid proxy key D_{ps} to A_2.
- **PSign Query**: On input of a message m, identities ID_U,ID_V and warrant m_w, the challenger C first executes the **ProxyKeyGen** and **CertGen** algorithms to access the proxy singing key, then executes the **PSign** algorithm and generates the valid proxy signature S and returns it to A_2.
- **Forge**: Finally, A_2 outputs a signature S^* with the warrant m_w^*, ID_V and the message m^* such that:
  1- S^* is a valid CBPS on the message m^* under the public key PK_V^* and the warrant message m_w^* where PK_V is output from the **UserKeyGen** query.
  2- ID_V has not been requested as the **Private Key** query.
  3- m_w^*, ID_V and ID_U, has not been requested as one of the **ProxyKeyGen** queries.
  4- The tuple (m^*, m_w^*, ID_V, ID_U) has not been requested as one of the **PSign** queries.

Definition 2
An adversary A_2 is called a $(\epsilon, t, q_c, q_s, q_H)$-forger if it has the advantage of at least ϵ in the above game which runs in time at most t and makes at most q_c, q_s and q_H create, signing and random oracle queries, respectively. Hence, a scheme is called $(\epsilon, t, q_c, q_s, q_H)$-secure against A_2 or actually unforgeable against chosen message an ID attack if no $(\epsilon, t, q_c, q_s, q_H)$-forger exists [4].

Definition 3
A challenger C is called an (ϵ', t')-solver if it can solve the ECDLP (Elliptic Curve Discrete Logarithm Problem) with unnegligible probability at least ϵ'in time at most t'.

3. Proposed scheme

In the present section, the first CBPS scheme with message recovery over ECC without pairings is proposed. Our scheme is inspired by [4] and [18]. The security of scheme is based on ECDLP assumption and will be proved in random oracle model. The proposed scheme consists of four entities: certification authority (CA), original signer, proxy signer, verifier and six algorithms: Setup, UserKeyGen, CertGen, ProxyKeyGen, PSign and PSVerif/MRecovery. In the scheme, we apply the symbol $E(a', b')$which denotes an elliptic curve over a prime finite field F_p. Let G be a cyclic subgroup of the elliptic curve group $E(a', b')$with generator P of order n. Our proposed scheme is based on the ECDLP assumption, i.e. for given Q in G, computing s_C such that $Q = s_C P$ is intractable [4].

Setup
CA gets a security parameter δ and returns system parameters $param$ as follows:

1) Chooses a δ-bit prime p and chooses the tuple $\{F_P, E(a', b'), G, P\}$ defined as above.
2) Chooses the master private key, $s_C \in_R Z_q^*$, and computes the master public key $PK_C = s_C P$.
3) Chooses secure cryptographic hash functions $H_i: \{0,1\}^* \times G^3 \rightarrow Z_n^*$, $i = 1,2$, $H_3: \{0,1\}^* \times G^2 \rightarrow Z_n^*$ and $H: \{0,1\}^* \rightarrow Z_n^*$.
4) Finally publishes tuple $param = \{F_P, E(a', b'), G, P, PK_C, H_1, H_2, H_3, H\}$ as system parameters and keeps s_C secret as the master private key.

UserKeyGen

Each user U with identity ID_U randomly picks $s_U \in Z_n^*$ and using system parameters $param$, generates his/her public/private key pairs $(PK_U, SK_U) = (s_U P, s_U)$ respectively (during the scheme, ID_A, ID_B represent the identity of original signer and proxy signer respectively).

CertGen

1) User U, transmits his/her public key and identity via an authenticated channel to the CA.
2) CA chooses at random $k_U \in_R Z_n^*$ and computes tuple $Cert_U = (d_U, U_U)$ as certificate of user U as follows. CA transmits user certificate to his/her via an authenticated channel.

$$U_U = k_U P \quad (1)$$

$$h_U = H_1(i, ID_U, PK_U, PK_C, U_U) \quad (2)$$

$$d_U = k_U + s_C h_U \ mod\ n \quad (3)$$

3) User U can validate its certificate by checking whether the following holds:

$$d_U P = U_U + h_U PK_C \quad (4)$$

ProxyKeyGen

1) The original signer A, after generation the warrant m_w, chooses at random $a \in_R Z_n^*$ and computes delegation W as follows and transmits tuple (m_w, U_A, K, W) to his/her proxy signer, B.

$$K = aP \quad (5)$$

$$e_1 = H_2(m_w, ID_B, K, U_A, PK_A) \quad (6)$$

$$W = (a + e_1(d_A + s_A)\ mod\ n) \quad (7)$$

2) The proxy signer B, after receiving the delegation (m_w, U_A, K, W), first computes h_A and e_1 values using equations (2) and (6) then checks whether the relation (8) holds.

$$WP = K + e_1(U_A + h_A PK_C + PK_A) \quad (8)$$

3) If the equation (8) is satisfied, proxy signer B, computes proxy key D_{ps} as follows:

$$e_2 = H_2(m_w, ID_A, K, U_B, PK_B) \quad (9)$$

$$D_{ps} = \left(W + e_2(d_B + s_B)\right) mod\ n \quad (10)$$

PSign

In this step, proxy signer B, computes his signature on the message $m \in \{0,1\}^*$ using system parameters $param$ and proxy key D_{ps} as follows:

1) Chooses at random $b \in_R Z_n^*$ and computes following value.

$$R = bP \quad (11)$$

2) Computes parameter r from equation (12). Where $(R)_x$ denotes the first coordinates of point R from elliptic curve group.

$$r = (m||H(m) + (R)_x) \quad (12)$$

3) Computes the value e and signature S as follows:

$$e = H_3(r, ID_A, ID_B, U_A, U_B) \quad (13)$$

$$S = (b - eD_{ps}) \quad (14)$$

4) The proxy signature will be the tuple $(m_w, W, U_A, U_B, K, r, S)$.

PSVerif/MRecovery

In order to verify the signature $(m_w, W, U_A, U_B, K, r, S)$ on the message m, the verifier acts as follows:

1) Check authorization of the proxy signer B in the warrant m_w and W.
2) Computes h_A ،h_B ،e_1 ،e_2 values from equations (2, 6 and 9).
3) Computes the $m||H(m)$ value from equation (15).

$$m||H(m) = r - \left(S.P + e\left(e_1(U_A + h_A PK_C + PK_A) + K + e_2(U_B + h_B PK_C + PK_B)\right)\right)_x \quad (15)$$

4) The correctness of the message and the proxy signature is verified if the hash result of the first part of $m||H(m)$ is equal to the second part of $m||H(m)$.

Correctness

According to the equations (10, 11, and 14) we have:

$$S.P = bP - eD_{ps}.P = R - e\left(D_{ps}.P\right) = R - e\left(WP + e_2(d_B P + s_B P)\right) \quad (16)$$
$$= R - e\left(e_1(U_A + h_A PK_C + PK_A) + K + e_2(U_B + h_B PK_C + PK_B)\right)$$

On the other, according to the equation (13), the value r is: $r = (m||H(m) + (R)_x)\ mod\ n$. Accordingly, we have:

$$m||H(m) = r - (R)_x \quad (17)$$
$$= r - (S.P + e(e_1(U_A + h_A PK_C + PK_A) + K + e_2(U_B + h_B PK_C + PK_B)))_x$$

To convert this scheme to a CBPS scheme without bilinear pairing and without message recovery, the **PSign** and **PSVerif** steps should be modified as follows:

PSign

In this step, proxy signer B, computes his signature on the message $m \in \{0,1\}^*$ using system parameters $param$ and proxy key D_{ps} as follows:

1) Chooses at random $b \in_R Z_n^*$ and computes following values:

$$R = bP \tag{18}$$

$$h = H_3(m, R) \tag{19}$$

2) Checks whether the equation (20) holds. Continue if it holds and otherwise return to step 1.

$$gcd(b + h, n) = 1 \tag{20}$$

5) Computes parameter s from equation (23).

$$s = (b + h)^{-1} D_{ps} \ mod\ n \tag{21}$$

6) The proxy signature will be the tuple $(m_w, W, U_A, U_B, K, R, s)$.

PSVerify

In order to verify the signature $(m_w, W, U_A, U_B, K, R, s)$ on the message m, the verifier does as follows:

1) Checks authorization of the proxy signer B in the warrant m_w and W.
2) Computes h, h_A ،h_B ،e_1 ،e_2 values by aforementioned equations.
3) Checks whether the equation (22) holds. Accepts signature if it holds and reject it otherwise.

$$s(R + h.P) = \mathrm{e}_1(U_A + h_A PK_C + PK_A) + e_2(U_B + h_B PK_C + PK_B) + K \tag{22}$$

Correctness

According to equation (18, 21) we have:

$$\begin{aligned} s(R + h.P) = (b + h)^{-1} D_{ps}(bP + h.P) = D_{ps}.P &= \left(W + e_2(d_B + s_B)\right) P \\ &= (a + e_1(d_A + s_A)P + e_2(d_B + s_B)P \\ &= K + \mathrm{e}_1(U_A + h_A PK_C + PK_A) + e_2(U_B + h_B PK_C + PK_B) \end{aligned} \tag{23}$$

4. Security Analysis of the Scheme

In this section, we will prove that our scheme is secure against adaptive chosen message and ID attack in the random oracle model under the difficulty of resolving ECDLP.

Theorem. The proposed scheme is $(\epsilon, t, q_c, q_s, q_H)$-secure against A_I and A_{II} adversaries in the random oracle model [4].

Proof. Suppose that A_1 or A_2 is $(\epsilon, t, q_c, q_s, q_H)$-adversary such that can forge the proposed certificate-based proxy signature with probability greater than ϵ. In this case, it will be shown that, the challenger C would be a $(\epsilon', t') - ECDLP$ solver whereas ϵ′ and t′ are calculated from the following equations [4]:

$$\epsilon' = \left(1 - \frac{q_H(q_c + q_s)}{n}\right)\left(1 - \frac{1}{n}\right)\left(\frac{1}{q_H}\right)^3 \epsilon \quad (24)$$

$$t' = t + (q_c + q_s)T \quad (25)$$

Proof against A_1. Suppose that the challenger C is tasked to solve discrete logarithm problem on the elliptic curves (ECDLP). Hence, for example if P and Q are two points on the elliptic curve such that $Q = s_C P$, the purpose of the challenger is computing s_C. In order to do this, C picks two identities ID_A and ID_B randomly as the challenged identities in the following game and gives $param = \{F_P, E(a', b'), G, P, PK_C = Q, H_1, H_2, H_3, H\}$ to A_1. Afterwards, C responds A_1 queries as follows:

1) **UserKeyGen query**: At first, list L_K, including diverse tuple $\{ID_U, SK_U, PK_U\}$, is empty. When A_1 queries on ID_U,C searches L_K for this input. If it is founded, C returns the public key PK_U otherwise C randomly chooses a_U for identity ID_U and computes $SK_U = a_U$ and $PK_U = a_U P$. Else, C adds the tuple $\{ID_U, SK_U, PK_U\}$ to L_K and returns PK_U to A_1.
2) $\boldsymbol{H_1}$ **query**: Firstly, the list L_{H_1}, including diverse tuple $\{i, ID_U, PK_U, PK_C, U_U\}$, is empty. When A_1 queries H_1 on ID_U, C randomly chooses $c_U \in_R Z_n^*$ and sets $h_U = H_1(i, ID_U, PK_U, PK_C, U_U) \leftarrow (-c_U)$ then C adds the tuple $\{i, ID_U, PK_U, PK_C, U_U, h_U\}$ to L_{H_1} and returns h_U to A_1.
3) $\boldsymbol{H_2}$ **query**: At first, the list L_{H_2}, including diverse tuple $\{m_w, K, U_U, PK_U, e_1, e_2\}$, is empty. When A_1 queries H_2 on ID_U and ID_V (ID_V is proxy signer identity and ID_U is original signer identity), C randomly chooses numbers $e_1, e_2 \in_R Z_n^*$ and sets $H_2(m_w, ID_V, K, U_U, PK_U) \leftarrow e_1$ and $H_2(m_w, ID_U, K, U_V, PK_V) \leftarrow e_2$. Then, C adds the tuples $\{m_w, ID_V, K, U_U, PK_U, e_1\}$ and $\{m_w, ID_U, K, U_V, PK_V, e_2\}$ to L_{H_2} and returns e_1, e_2 to A_1.
4) $\boldsymbol{H_3}$ **query**: First of all, the list L_{H_3}, including diverse tuple $\{m, ID_U, ID_V, r, U_U, U_V, e\}$, is empty. When A_1 queries H_3 on message m, C randomly chooses number $e \in_R Z_n^*$ and sets $H_3(r, ID_U, ID_V, U_U, U_V) \leftarrow e$. Then C adds the tuple $\{m, ID_U, ID_V, r, U_U, U_V, e\}$ to L_{H_3} and returns e to A_1.
5) **PrivateKey query**: For input ID_U, C first checks whether $ID_U = ID_B$ or $ID_U = ID_A$, and if so, stops. Else, C searches the tuple $\{ID_U, SK_U, PK_U\}$ in the L_k and returns SK_U to A_1.
6) **ReplaceKey query**: For input ID_U and public/private key pair (SK_U, PK_U), C searches L_K for identity ID_U and updates the tuple $\{ID_U, SK_U, PK_U\}$ by (ID_U, SK_U^*, PK_U^*).
7) **CertGen query:** At first, the list L_{Ce}, including diverse tuple $\{ID_U, PK_U, Cert_U\}$, is empty. When A_1 queries certificate generation on ID_U and PK_U, C first checks whether $ID_U = ID_B$ or $ID_U = ID_A$, and if so, stops. Else, randomly chooses value $b_U \in_R Z_n^*$ and sets $d_U \leftarrow b_U$ and $U_U \leftarrow b_U P + c_U PK_C$. Then, C adds the tuples $\{ID_U, PK_U, Cert_U\}$ to L_{Ce} and return certificate $Cert_U = (d_U, U_U)$ to A_1.

8) **ProxyKeyGen query**: At first, the list L_{pk}, including diverse tuple $\{m_w, ID_U, ID_V, a, K, D_{ps}\}$, is empty. When A_1 queries proxy key generation on ID_U, ID_V and warrant m_w, C simulates the oracle as follows:
   - If the tuple $\{m_w, ID_U, ID_V, *, *, *\}$ is in the L_{pk}, then C returns the D_{ps}.
   - Otherwise, if $ID_U = ID_B$ or $ID_V = ID_A$ or $ID_V = ID_B$ or $ID_U = ID_A$, C will terminate to the simulation of the oracle. Else, C looks up the tables L_K and L_{Ce} for private keys and certificates of the ID_U, ID_V then chooses randomly $a \in_R Z_n^*$ and computes the following parameters:

$$K = aP \quad (26)$$

$$e_1 = H_2(m_w, ID_V, K, U_U, PK_U) \quad (27)$$

$$W = (a + e_1(d_U + s_U)\ mod\ n) \quad (28)$$

$$e_2 = H_2(m_w, ID_U, K, U_V, PK_V) \quad (29)$$

$$D_{ps} = \big(W + e_2(d_V + s_V)\big)\ mod\ n \quad (30)$$

Then C returns D_{ps} to A_1 and adds the tuples $\{m_w, ID_U, ID_V, a, K, D_{ps}\}$ to the L_{pk} and $\{m_w, ID_V, K, U_U, PK_U, e_1\}$, $\{m_w, ID_U, K, U_V, PK_V, e_2\}$ to the L_{H_2}.

9) **PSign query**: For input the message m, identity pairs $\{ID_U, ID_V\}$ and warrant m_w, the challenger C first checks L_{pk} for tuple $\{m_w, ID_U, ID_V, *, *, *\}$ if that is fined, C signs the message m with the proxy signature key D_{ps}. Otherwise, C does similar in the **ProxyKeyGen** algorithm in the scheme and computes the proxy key D_{ps} then uses it to sign the message m. Eventually, C adds the tuple $\{m, ID_U, ID_V, r, U_U, U_V, e\}$ to the L_{H_3} and returns the proxy signature to A_1.

Finally, if A_1 can forge a valid proxy signature $(m_w^*, W^*, U_A, U_B, K, r, S^{(1)})$ on the message m^* such that:

- B was not designated by A as a proxy signer
- The identities ID_A and ID_B have not been submitted at the same time to **PrivateKeyGen** oracle and **ReplaceKey** oracle.
- The identities ID_A and ID_B are not requested as input to the **PrivateKeyGen** oracle and **CertGen** oracle.
- ID_A, ID_B with warrant m_w^* are not requested as input to the **ProxyKeyGen** oracle.
- The message m^* with m_w^* and identities ID_A, ID_B, are not requested as input to the **PSign** oracle.

In this case, C performs the game with A_1 for six more times again with variables e_1 ،e_2 and e and constant parameters K, r. At each performance, only $e_1^{(i)}$, $e_2^{(i)}$ and $e^{(i)}$ are changed which are outputs of random oracles $H_2(m_w^*, ID_B, K, U_A, PK_A)$,

$H_2(m_w^*, ID_A, K, U_B, PK_B)$ and $H_3(r, ID_A, ID_B, U_A, U_B)$. Therefore, for $i = 1, \dots, 7$ we will have:

$$S^{(i)}.P = R - e^{(i)}\left(e_1^{(i)}(U_A + h_A PK_C + PK_A) + K + e_2^{(i)}(U_B + h_B PK_C + PK_B)\right) \quad (31)$$

It is clear that the values $b, s_A, s_B, k_A, k_B, x_C, a$ are elliptic curve discrete logarithm of points $R, PK_A, PK_B, U_A, U_B, PK_C, K$ respectively such that, $R = bP$, $PK_A = s_A P$, $PK_B = s_B P$, $U_A = k_A P$, $U_B = k_B P$, $PK_C = s_C P$ and $K = aP$. According to the equation (31), seven linearly independent equations apparently for $i = 1, \dots, 7$ will be provided as:

$$S^{(i)} = b - e^{(i)}\left(e_1^{(i)}(k_A + h_A s_C + s_A) + a + e_2^{(i)}(k_B + h_B s_C + s_B)\right) \ mod\ n \quad (32)$$

Due to the fact that only values $b, s_A, s_B, k_A, k_B, x_C, a$ are unknown to the C, then there will exist a simple algorithm to solve these seven linearly independent equations with seven unknowns and outputs s_C as a solution of ECDLP.

Proof against A_2. This proof is similar to the proof for the adversary type one. This means that the game performs between the challenger C and A_2. The adversary A_2 can request all above queries except the **ReplaceKey** query of the challenger C. Moreover, since the master private key s_C is available for A_2, it won't require to **CertGen** query. Eventually, if A_2 can forge a valid proxy signature $(m_w^*, W^*, U_A, U_B, K, r, S^{(1)})$ on the message m^* such that:

- B was not designated by A as a proxy signer
- The identities ID_A and ID_B are not requested as input to the **ReplaceKey** oracle.
- The identities ID_A and ID_B are not requested as input to the **PrivateKeyGen** oracle.
- ID_A, ID_B with warrant m_w^* are not requested as input to the **ProxyKeyGen** oracle.
- The message m^* with m_w^* and identities ID_A, ID_B, are not requested as input to the **PSign** oracle.

In this case, C performs the game with A_2 for more six times again with variables e_1, e_2, and e and constant parameters K, r. Similar to the presentation of security proof against A_I, with using achieved seven linearly independent equations apparently, it will exist a simple algorithm to solve the seven linearly independent equations with seven unknowns and outputs s_C as a solution of ECDLP. If the seven independent equations aren't linear, this process is repeated up to the attainment of seven linearly independent equations.

Advantage

In response to the **CertGen** query, if C assigns the conflicting value for random oracle $\mathrm{H}_1(\mathrm{i}, \mathrm{ID_U}, \mathrm{PK_U}, \mathrm{PK_C}, \mathrm{U_U})$, the simulation of certificate generation oracle fails. This case happens with probability at most $(\frac{q_H}{n})$. Since there are at most q_c certificate generation queries and q_s signing queries, so simulation of certificate generation oracle would be successful at most $(q_c + q_s)$ times with probability at least $(1 - \frac{q_H(q_c+q_s)}{n})$. Furthermore, it

is assumed that random oracles have ideal randomness behavior. Therefore, according to the three queries $H_2(m_w, ID_V, K, U_U, PK_U)$, $H_2(m_w, ID_U, K, U_V, PK_V)$ and $H_3(r, ID_U, ID_V, U_U, U_V)$ with probility at least $\left(1 - \frac{1}{n}\right)$, C makes a correct response of such a query at the point of representation with probability at least $\left(\frac{1}{q_H}\right)^3$. Thus, the overall success probability will be at least $\epsilon' = \left(1 - \frac{q_H(q_c+q_s)}{n}\right)\left(1 - \frac{1}{n}\right)\left(\frac{1}{q_H}\right)^3 \epsilon$ and the overall time to complete the algorithm, dominated by the exponentiations performed in the certificate generation and signing queries, which is equal to $t' = t + (q_c + q_s)T$ [4].

5. Conclusion

With respect to the performance of the certificate-based infrastructure in comparison with the other public key infrastructures in addition to the performance of the schemes without bilinear pairing compared to the other similar schemes, a certificate-based proxy signature scheme without bilinear pairing will be an efficient one. In the present study, a certificate-based proxy signature scheme without bilinear pairing is proposed which is capable of message recovery; hence, the message doesn't need to be sent along with the signature, which saves storage space and communication bandwidth. The security of the scheme is proved in the random oracle model and it has been proved that the presented scheme of this study is secure against adaptive chosen message and ID attack for two types of the adversaries.

References

[1] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: Delegation of the power to sign messages," IEICE transactions on fundamentals of electronics, communications and computer sciences, vol. 79, pp. 1338-1354, 1996.

[2] B. Lee, H. Kim, and K. Kim, "Strong proxy signature and its applications," in Proc of SCIS, pp. 603-608, 2001.

[3] K Nyberg, RA Rueppel. "Message recovery for signature schemes based on the discrete logarithm problem." Advances in Cryptology—EUROCRYPT'94. Springer Berlin Heidelberg, 1995.

[4] S. Padhye and N. Tiwari, "ECDLP-based certificateless proxy signature scheme with message recovery," Transactions on Emerging Telecommunications Technologies, 2012.

[5] A. Shamir, "Identity-based cryptosystems and signature schemes," in Advances in Cryptology-CRYPTO'84, pp. 47-53, 1985.

[6] C. Gentry, "Certificate-based encryption and the certificate revocation problem," Lecture notes in computer science, vol. 2656, pp. 272-293, 2003.

[7] S. S. AL-Riyami and K. G. Paterson, "Certificateless public key cryptography," Lecture notes in computer science, pp. 452-473, 2003.

[8] X Huang, W Susilo, Y Mu, F Zhang. "On the security of certificateless signature schemes from Asiacrypt 2003." In Cryptology and Network Security, pp. 13-25. Springer Berlin Heidelberg, 2005.

[9] BC Hu, DS Wong, Z Zhang, X Deng. "Key replacement attack against a generic construction of certificateless signature." In *Information Security and Privacy*, pp. 235-246. Springer Berlin Heidelberg, 2006.

[10] J. K. Liu, J. Baek, W. Susilo, and J. Zhou, "Certificate-Based Signature Schemes without Pairings or Random Oracles," in Proceedings of the 11th international conference on Information Security, pp. 285-297, 2008.
[11] J. Li, L. Xu, and Y. Zhang, "Provably secure certificate-based proxy signature schemes," Journal of Computers, vol. 4, pp. 444-452, 2009.
[12] B. G. Kang, J. H. Park, and S. G. Hahn, "A certificate-based signature scheme," Lecture notes in computer science, pp. 99-111, 2004.
[13] J. Li, X. Huang, Y. Mu, W. Susilo, and Q. Wu, "Certificate-based signature: security model and efficient construction," in Proceedings of the 4th European conference on Public Key Infrastructure: theory and practice, pp. 110-125, 2007.
[14] L Wang, J Shao, Z Cao, M Mambo, and A Yamamura. "A certificate-based proxy cryptosystem with revocable proxy decryption power." In Progress in Cryptology–INDOCRYPT 2007, pp. 297-311. Springer Berlin Heidelberg, 2007.
[15] XQ Liang, SD Liu, JF Xu, Q Liu. "A certificate-based strong designated verifier proxy signature scheme." In Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, vol. 1, pp. V1-614. IEEE, 2010.
[16] H. Singh and G. K. Verma, "ID-based proxy signature scheme with message recovery," Journal of Systems and Software, vol. 85, pp. 209-214, 2012.
[17] M. Tian, L. Huang, and W. Yang, "Cryptanalysis of an ID-based proxy signature scheme with message recovery," Appl. Math, vol. 6, pp. 419-422, 2012.
[18] H. Debiao, C. Jianhua, and H. Jin, "An ID-based proxy signature schemes without bilinear pairings," annals of telecommunications-annales des telecommunications, vol. 66, pp. 657-662, 2011.