International Journal of Computer Networks and Communications Security

VOL. 3, NO. 2, FEBRUARY 2015, 33–42
Available online at: www.ijcncs.org
E-ISSN 2308-9830 (Online) / ISSN 2410-0595 (Print)

Utilization of ECDLP for Constructing a New Certificate Based Digital Signature

Leili Abedi-Ostad[1] and Morteza Nikooghadam[2]

[1, 2] Department of Computer Engineering, University of Imam Reza, Mashhad 91735-553, Iran

E-mail: [1]leilaabediostad@imamreza.ac.ir, [2]m.nikooghadam@imamreza.ac.ir

ABSTRACT

Digital signatures that are used to achieve the integrity along with the authentication could be classified into various types. PKI based, ID based, certificate based and certificateless digital signatures are the most important types. Regarding advantages of certificate-based signatures (CBS), we want to propose a CBS scheme by means of employing elliptic curve discrete logarithm problem (ECDLP).The proposed scheme's security is proven under the elliptic curve discrete logarithm assumption in the random oracle model. Results of comparing our scheme with existing pairing-free certificate-based signature schemes, shows that ours has much lower computational cost.

Keywords: *Elliptic Curve Discrete Logarithm Problem, Certificate-Based Digital Signature, Random Oracle Model, Pairing-Free, Elliptic Curve Cryptography.*

1 INTRODUCTION

In traditional public key cryptography (PKC), the public key of user should be certified by certification authority (CA). This approach has difficulties for managing certificates. For solving this problem, Shamir [1] presented identity based cryptography (IBC), which means user's public key is made of his/her unique identity. In this scheme, private key of user is created by a private key generator (PKG). Since PKG has private key of all users, he/she can impersonate them. This problem is called key escrow [2].

There are two schemes for solving this problem. One of them is certificateless public key cryptography (CL-PKC). This scheme was presented by Al-Riyami and Paterson [3]. In this scheme, a key generating center (KGC) has to create user's partial private key. The private key is made of the partial private key and a random secret value that is selected by the user. Since users select their own public key, there is no way for authentication of declared public key. This problem leads to key replacement attack [4]. The other scheme is certificate-based cryptography (CBE) which was presented by Gentry [2]. In this scheme all users make their private and public key. Afterwards CA produces a certificate for each user by using his/her identity and public key.

Certificate-based signature (CBS) was proposed by Kang et al. [5]. In this scheme, similar to CBE, private and public keys are created by the user; then CA creates a certificate based on user's public key and his/her identity. A signer by knowing his/her certificate and private key can produce a valid signature. In [4-10] many certificate-base signatures based on pairing operation were proposed. In 2000, Koblitz et al. [11] found out that the computational cost of exponentiation operation is much more than the cost of scalar multiplication on the elliptic curve group. In 2007, Chen et al. [12] realized that the computational cost of pairing is twenty times more than scalar multiplication over the elliptic curve group. Since cryptography protocols without pairing have much lower cost than pairing-based protocols, Liu et al. [13] suggested one pairing-free CBS schemes. Zhang [14] demonstrated that the proposed pairing-free CBS scheme in [13] was insecure. In 2009, Ming and Wang [15] and Zhang et al. [16] suggested schemes without pairing. Li et al. [10] suggested two secure CBS schemes against key replacement attack. In 2012, Li et al. [17] suggested a short CBS scheme which had one pairing operation. Li et al. [18] in 2013 proposed a new CBS scheme under the

discrete logarithm assumption and secure in random oracle model.

We want to propose a CBS scheme by employing ECDLP. We will show that our scheme is secure under the elliptic curve discrete logarithm assumption in the random oracle model. Compared to existing CBS schemes, ours has much lower computational cost.

At first, we give some definitions. Then you can see our suggested CBS scheme and its security analysis. Efficiency comparison of our scheme and conclusion are at the end of this paper.

2 STRUCTURE OF CBS

Setup: It gets a security parameter, and gives the system public parameters and the certifier's master secret key.

UserKeyGen: It gets the system public parameters, and gives a secret key and a public key.

Certify: It gets system public parameters, master secret key, the identity of a user and its public key. Then its output is the user certificate.

Sign: It gets system public parameters, a message, the user's identity and his/her certificate, public key and secret key. Its output is a signature.

Verify: It gets a message/signature pair, system parameters, user's public key and his/her identity. Its output is 0 or 1. Value 1 indicates a valid signature, and 0 is for an invalid signature.

3 SECURITY MODEL

According to [4, 5 and 18], we should consider adversary and adversary. Adversary is a malicious user who can be anyone except the CA. He can't gain the certificate of the other users but he can change their public keys. He can't gain the CA's master secret key, either. Adversary is a malicious CA who has a master secret key but is not able to change the user's public key. We use the same security model in [18] for analyzing security of proposed scheme.

4 SUGGESTED CBS SCHEME

Setup*:* This algorithm gets security parameter *k* and outputs system parameters and master key. CA proceeds as follows:

Selects a *k*-bit prime p and determines the tuple $\{F_p, \frac{E}{F_p}, G, P, H_0, H_1, H_2\}$. Selects the master private key $x \in Z_n^*$ and calculates the master public key $y = x.P$. Selects three cryptographic secure hash functions $H_0 : \{0,1\}^* \times G \times G \to Z_n^*$, $H_1 : \{0,1\}^* \times G \times G \times G \to Z_n^*$ and $H_2 : \{0,1\}^* \times \{0,1\}^* \times G \times G \times G \to Z_n^*$.

Publishes $\{F_p, \frac{E}{F_p}, G, P, Q_c, H_0, H_1, H_2\}$ as system parameters and preserves the master key *x*.

UserKeyGen: This algorithm gets parameters, chooses $x_{ID} \in Z_n^*$ randomly as the user private key and then calculates $PK_{ID} = x_{ID}.P$ as the user public key.

Certify: This algorithm gets parameters, master secret key *x*, user public key PK_{ID} and user identity $ID \in \{0,1\}^*$. Randomly picks $s \in Z_n^*$ and computes $W = s.P$, $h_0 = H_0(ID, PK_{ID}, W)$ and $R = s + x.h_0 \mod n$. The output is the user's certificate $Cert_{ID} = \langle R, W \rangle$.

User will validate his/her certificate by checking the equation $R.P = W + y.H_0(ID, PK_{ID}, W)$.

Sign: It gets parameters, user identity *ID*, user private key x_{ID}, user certificate $Cert_{ID}$ and message $m \in \{0,1\}^*$. The algorithm works as follows: Chooses $r \in Z_n^*$ randomly and computes $U = r.P$. Calculates $h_1 = H_1(m, PK_{ID}, U, W)$ and $h_2 = H_2(m, ID, PK_{ID}, U, W)$.

Calculates $z = (R + x_{ID}.h_1 + r.h_2) \mod n$.

The signature is $\sigma = \langle U, W, z \rangle$.

Verify: Takes parameters, user public key PK_{ID} and message/signature pair (m, σ) and computes $h_0 = H_0(ID, PK_{ID}, W)$, $h_1 = H_1(m, PK_{ID}, U, W)$, $h_2 = H_2(m, ID, PK_{ID}, U, W)$.

This algorithm examines the equation:

$$z.P = W + y.h_0 + PK_{ID}.h_1 + U.h_2 \quad (1)$$

If the equality holds, the output is 1; if not, the output is 0.

The reason that the verification equation holds for valid signatures is:

$$\begin{aligned} & W + y.h_0 + PK_{ID}.h_1 + U.h_2 \\ &= s.P + x.h_0.P + x_{ID}.h_1.P + r.h_2.P \\ &= (s + x.h_0).P + x_{ID}.h_1.P + r.h_2.P \\ &= R.p + x_{ID}.h_1.P + r.h_2.P \\ &= (R + x_{ID}.h_1 + r.h_2).P \\ &= z.P \end{aligned} \quad (2)$$

In Figure 1, Setup, UserKeyGen and Certify steps and in Figure 2, Sign and Verify steps are shown.

Signer		***CA***
		(1) Chooses $x \in Z_n^*$ randomly Computes $y = x.P$ Chooses H_0, H_1 and H_2
(2) Selects $x_{ID} \in Z_n^*$ randomly Computes $PK_{ID} = x_{ID}.P$	PK_{ID} and $ID \in \{0,1\}^*$ →	Publishes $\{F_p, \frac{E}{F_p}, G, P, y, H_0, H_1, H_2\}$
		(3) Randomly picks $s \in Z_n^*$ Computes $W = s.P$, Computes $h_0 = H_0(ID, PK_{ID}, W)$
	← $Cert_{ID} = \langle R, W \rangle$	Computes $R = s + x.h_0 \bmod n$
(4) Certificate Verification: $R.P = W + y.H_0(ID, PK_{ID}, W)$		

Fig. 1. Interactions between the signer and CA

Signer		***Verifier***
(1) Chooses $r \in Z_n^*$ randomly Computes $U = r.P$ Computes $h_1 = H_1(m, PK_{ID}, U, W)$ and $h_2 = H_2(m, ID, PK_{ID}, U, W)$ $z = (R + x_{ID}.h_1 + r.h_2) \bmod n$ The signature is $\sigma = \langle U, W, z \rangle$	PK_{ID} and (m, σ) →	(2) Computes $h_0 = H_0(ID, PK_{ID}, W)$ Computes $h_1 = H_1(m, PK_{ID}, U, W)$ Computes $h_2 = H_2(m, ID, PK_{ID}, U, W)$ Signature Verification: $z.P = W + y.h_0 + Pk_{ID}.h_1 + U.h_2$

Fig. 2. Interactions between the signer and the verifier

5 SECURITY ANALYSIS

Theorem 1: (Game I). Let A_I be a Type I adversary against proposed CBS scheme in random oracle model and runs at most *t* in polynomial time, makes at most q_{H_0} (for *i* = 0,1,2) H_i queries, q_r PKReplace queries, q_e certification queries, q_c corruption queries, q_k UserKeyGen queries and q_s sign queries and wins the **Game I** with a probability ε. An algorithm *B* can solve the ECDLP with a probability ε' in polynomial time t', where

(3)

$$\varepsilon' \geq \frac{1}{q_{H_0}}\left(1-\frac{1}{q_{H_0}}\right)^{q_e+q_s}\varepsilon,$$

$$t' \leq 2t + (q_k + 2q_e + 4q_s)t_e + (q_e + q_s)t_m$$

The multiplication operation in Z_n^* takes time t_e and addition operation in $i = j$ takes time t_m in the random oracle model.

Proof: Let $(F_p, \frac{E}{F_p}, G, P, Q = \alpha.P)$ be a random instance of the ECDLP selected by *B* as input. *B* wants to output α. Hash functions are considered as random oracles. For consistence, *B* requires keeping five initially empty lists L_k, L_e, L_0, L_1, L_2. List L_k keeps the UserKeyGen queries and PKReplace queries; list L_e keeps certification queries and lists L_0, L_1, L_2 keep H_i queries. At first *B* sets the master public key $y = Q = \alpha.P$ and gives system parameters $F_p, \frac{E}{F_p}, G, P, y$ to A_I. Then, *B* randomly selects an index *j* such that $1 \leq j \leq q_{H_0}$, where q_{H_0} is the number of queries in

the random oracle H_0. We note that first $ID_j = ID^*$ where ID_j is the j^{th} query to the random oracle H_0 and j should be selected. Algorithm B will simulate oracles and interact with the adversary A_I as follows:

UserKeyGen Query: This algorithm gets a user's identity ID_i. Then B verifies the list L_k to see whether ID_i has been inserted before or not. If it was not defined, B selects $x_{ID_i} \in Z_n^*$ randomly and puts $PK_{ID_i} = x_{ID_i}.P$. Then B adds $(ID_i, x_{ID_i}, PK_{ID_i})$ to the list L_k and transfers PK_{ID_i} to A_I. Otherwise, it sends back the defined value.

H_0 **Query**: This algorithm gets (ID_i, PK_{ID_i}, W_i), Then B verifies the list L_0 to see whether H_0 has been inserted before for that input or not. If it was not defined, B selects $d_i \in Z_n^*$ randomly and sends it back as a hash value of (ID_i, PK_{ID_i}, W_i). Then B adds $(ID_i, PK_{ID_i}, W_i, d_i)$ to the list L_0. Otherwise, it sends back the defined value.

H_1 **Query**: Gets $(m_i, PK_{ID_i}, U_i, W_i)$. Then B verifies the list L_1 to see whether H_1 has been inserted before for that input or not. If it was not defined, B selects $e_i \in Z_n^*$ randomly and sends it back as a hash value of $(m_i, PK_{ID_i}, U_i, W_i)$. Then B adds $(m_i, PK_{ID_i}, U_i, W_i, e_i)$ to the list L_1. Otherwise, it sends back the defined value.

H_2 **Query**: Gets $(m_i, ID_i, PK_{ID_i}, U_i, W_i)$. Then B verifies the list L_2 to see whether H_2 has been inserted before for that input or not. If it was not defined, B selects $c_i \in Z_n^*$ randomly and sends it back as a hash value of $(m_i, ID_i, PK_{ID_i}, U_i, W_i)$. Then B adds $(m_i, ID_i, PK_{ID_i}, U_i, W_i, c_i)$ to the list L_2. Otherwise, it sends back the defined value.

PKReplace Query: This algorithm gets a user's identity ID_i and public key PK'_{ID_i}, and then B verifies the list L_k to see if ID_i has been inserted before or not. If it was defined, B puts $PK_{ID_i} = PK'_{ID_i}$ and $x_{ID_i} = \perp$. Otherwise, B adds $(ID_i, \perp, PK'_{ID_i})$ to the list L_k.

Corruption Query: This algorithm gets user's identity ID_i, and then B verifies the list L_k to see if ID_i has been inserted before or not. If it was not defined, B selects $x_{ID_i} \in Z_n^*$ randomly and puts $PK_{ID_i} = x_{ID_i}.P$. Then B adds $(ID_i, x_{ID_i}, PK_{ID_i})$ to the list L_k and transfers x_{ID_i} to A_I. Otherwise, it sends back the defined value.

Certification Query: This algorithm gets ID_i and PK_{ID_i}, then B responds as follows:

If $i \neq j$, B verifies the list L_e to see whether ID_i has been inserted before or not. If not, B selects two random numbers d_i and $R_i \in Z_n^*$ and computes $W_i = R_i.P - d_i.y$. Then B verifies the list L_0 to see whether (ID_i, PK_{ID_i}, W_i) has been inserted before or not. If it was defined before, B must reselect d_i and $R_i \in Z_n^*$. Otherwise B adds $(ID_i, PK_{ID_i}, W, d_i)$ to the list L_0, adds $(ID_i, PK_{ID_i}, W_i, R_i)$ to list L_e and sends $Cert_{ID_i} = \langle W_i, R_i \rangle$ to A_I. Otherwise, it sends back the defined value. If $i = j$, B aborts.

Sign Query: This algorithm gets ID_i and m_i, then, B makes UserKeyGen query and Corruption query and gets PK_{ID_i} and x_{ID_i}. If $x_{ID_i} = \perp$, A_I should provide the matching secret key x_{ID_i}. Otherwise B responds as follows:

If $i \neq j$, B makes certification query and signs the message m_i by using $(Cert_{ID_i}, x_{IDi})$.

If $i = j$, B selects $e_j, c_j, z_j, d_i \in Z_n^*$ and computes $W_j = -d_j.y$ and $U_j = c_j^{-1}(z_j.P - PK_{ID_j}.e_j)$.

B sets $H_0(ID_j, PK_{ID_j}, W_j) = d_j$,

$H_1(m_j, PK_{ID_j}, U_j, W_j) = e_j$ and

$H_2(m_j, ID_j, PK_{ID_j}, U_j, W_j) = c_j$.

If hash functions H_0, H_1 and H_2 have been defined before, B reselects the random values. Otherwise, B adds $(ID_j, PK_{ID_j}, W_j, d_j)$ to the list L_0, adds $(m_j, PK_{ID_j}, U_j, W_j, e_j)$ to the list L_1 and adds $(m_j, ID_j, PK_{ID_j}, U_j, W_j, c_j)$ to the list L_2. Finally, (U_j, W_j, z_j) is given to A_I.

Therefore, A_I gives a forgery signature $\sigma^* = \langle U^*, W^*, z^* \rangle$ on message m^* by considering (ID^*, PK^*_{ID}). If $ID^* \neq ID_j$, B aborts. If not, by using the forking lemma [19], B repeats A_I with different oracle H_0 but the same random tape.

Then B can get another valid signature $\sigma' = \langle U^*, W^*, z' \rangle$. So,

$$z^*.P = W^* + y.h_0^* + PK_{ID}^*.h_1^* + U^*.h_2^* \quad (4)$$

$$z'.P = W^* + y.h_0' + PK_{ID}^*.h_1^* + U^*.h_2^* \quad (5)$$

From these two forgeries, B can compute $\alpha = \frac{z^* - z'}{h_0^* - h_0'}$, so B has solved the ECDLP. B can obtain the value of α if $P_r[E_1 \wedge E_2 \wedge E_3]$ where E_1: B does not fail while responding oracle queries, E_2: A_I wins and E_3: If $ID^* \neq ID_j$. From the simulation, we have $P_r[E_1] \geq \left(1 - \frac{1}{q_{H_0}}\right)^{q_e+q_s}$, $P_r[E_2|E_1] = \varepsilon$, $P_r[E_3|E_1 \wedge E_2] = \frac{1}{q_{H_0}}$ thus the success probability of B solving ECDLP is $\varepsilon' \geq \frac{1}{q_{H_0}}\left(1 - \frac{1}{q_{H_0}}\right)^{q_e+q_s} \varepsilon$. Algorithm B's running time t' is two times of the A_I's running time t and the time required to answer oracle queries and the time to solve the ECDLP. Totally B running time is $t' \leq 2t + (q_k + 2q_e + 4q_s)t_e + (q_e + q_s)t_m$. □

Theorem 2: (Game II). Let A_{II} be a Type II adversary against the proposed CBS scheme in random oracle model and wins the **Game II** with a probability ε. Then there is an algorithm B which can solve the ECDLP with a probability ε' in polynomial time t', where

(6)

$$\varepsilon' \geq \frac{1}{q_{H_0}}\left(1 - \frac{1}{q_{H_0}}\right)^{q_c+q_s} \varepsilon$$

$$t' \leq 2t + (q_k + q_c + 4q_s)t_e + (q_s)t_m$$

Proof: Let $(F_p, \frac{E}{F_p}, G, P, Q = \alpha.P)$ be a random instance of the ECDLP selected by B as input. B wants to output α. At first B selects $s \in Z_n^*$ randomly and sets master public key $y = s.P$ and gives system parameters $F_p, \frac{E}{F_p}, G, P, y$ and master secret key s to A_{II}. Then, B randomly picks an index j such that $1 \leq j \leq q_{H_0}$, where q_{H_0} is the number of queries to the random oracle H_0. It is noticeable that first $ID_j = ID^*$ where ID_j is the j^{th} query to the random oracle H_0 and j should be selected. Algorithm B will simulate oracles and interact with the adversary A_{II} as follows:

UserKeyGen Query: This algorithm gets a user's identity ID_i. Then B verifies the list L_k to see whether ID_i has been inserted before or not. If so, the defined value is sent back. If not, B responds as follows: If $i \neq j$ B chooses $x_{ID_i} \in Z_n^*$ randomly and sets $PK_{ID_i} = x_{ID_i}.P$. Then B adds $(ID_i, x_{ID_i}, PK_{ID_i})$ to the list L_k and transfers PK_{ID_i} to A_{II}. If $i = j$ B puts $PK_{ID_j} = Q$, then B adds $(ID_j, \perp, PK_{ID_j})$ to the list L_k and transfers PK_{ID_j} to A_{II}.

H_0, H_1 and H_2 queries are the same as H_0, H_1 and H_2 queries in theorem 1.

Corruption Query: This algorithm gets a user's identity ID_i, and then B responds as follows:

If $i \neq j$ B verifies the list L_k to see whether ID_i has been defined before or not. If it was not defined, B selects $x_{ID_i} \in Z_n^*$ randomly and puts $PK_{ID_i} = x_{ID_i}.P$. Then B adds $(ID_i, x_{ID_i}, PK_{ID_i})$ to the list L_k and transfers x_{ID_i} to A_{II}. Otherwise, it sends back the defined value. If $i = j$, B aborts.

Sign Query: this query is the same as sign query in theorem 1, but interacts with A_{II}.

Therefore, A_{II} gives a forgery signature $\sigma^* = \langle W^*, U^*, z^* \rangle$ on message m^* by considering (ID^*, PK_{ID}^*). If $ID^* \neq ID_j$, B aborts. If not, by using the forking lemma [19], B repeats A_{II} with different oracle H_1 but the same random tape. Then B can get another valid signature $\sigma' = \langle W^*, U^*, z' \rangle$. So,

$$z^*.P = W^* + y.h_0^* + PK_{ID}^*.h_1^* + U^*.h_2^* \quad (7)$$

$$z'.P = W^* + y.h_0^* + PK_{ID}^*.h_1' + U^*.h_2^* \quad (8)$$

Table 1: Time complexity comparison

Scheme	Sign generation phase	Time complexity in T_{Mul}	Verification phase	Time complexity in T_{Mul}
Scheme in [13]	$T_{EXP}+2\,T_{Mul}+T_{ADD}+T_{HASH}$	242 T_{Mul}	$7\,T_{EXP}+5\,T_{Mul}+3\,T_{HASH}$	1685 T_{Mul}
Scheme in [15]	$T_{EXP}+T_{Mul}+2\,T_{ADD}+T_{HASH}$	241 T_{Mul}	$3\,T_{EXP}+4\,T_{Mul}+2\,T_{HASH}$	724 T_{Mul}
Scheme in [16]	$3\,T_{EXP}+3\,T_{Mul}+3\,T_{ADD}+2\,T_{HASH}$	723 T_{Mul}	$7\,T_{EXP}+5\,T_{Mul}+4\,T_{HASH}$	1685 T_{Mul}
Scheme in [18]	$T_{EXP}+2\,T_{Mul}+2\,T_{ADD}+2\,T_{HASH}$	242 T_{Mul}	$4\,T_{EXP}+3\,T_{Mul}+3\,T_{HASH}$	963 T_{Mul}
Our scheme	$2\,T_{Mul}+2\,T_{ADD}+T_{EC-MUL}+2\,T_{HASH}$	31 T_{Mul}	$3\,T_{EC-ADD}+4\,T_{EC-MUL}+3\,T_{HASH}$	116.36 T_{Mul}

From these two forgeries, B can compute $\alpha=\frac{z^*-z'}{h_1^*-h_1'}$, so B has solved the ECDLP. B can obtain the value of α if $P_r[E_1\wedge E_2\wedge E_3]$ where E_1: B does not fail while responding oracle queries, E_2: A_{II} wins and E_3: If $ID^*=ID_j$. From the simulation, we have $P_r[E_1]\geq\left(1-\frac{1}{q_{H_0}}\right)^{q_c+q_s}$, $P_r[E_3|E_1\wedge E_2]=\frac{1}{q_{H_0}}$, $P_r[E_2|E_1]=\varepsilon$ thus the success probability of B solving ECDLP is $\varepsilon'\geq\frac{1}{q_{H_0}}\left(1-\frac{1}{q_{H_0}}\right)^{q_c+q_s}\varepsilon$. Algorithm B's running time t' is two times of the A_{II}'s running time t and the time required to respond oracle queries and the time to solve the ECDLP. Totally, B run time is $t'\leq 2t+(q_k+q_c+4q_s)t_e+(q_s)t_m$ □

6 EFFICIENCY COMPARISON

You can see the definition of used notations in this paper and their conversions in term of T_{Mul} in the following: [11, 20]

T_{Mul} is time complexity of performing a multiplication operation.

T_{EXP} is time complexity of performing an exponentiation operation. ($240T_{Mul}$)

T_{ADD} is Time complexity of performing an addition operation. (Negligible)

T_{EC-MUL} is time complexity of performing a multiplication of an elliptic curve point. ($29T_{Mul}$)

T_{EC-ADD} is time complexity of performing an addition of two points on elliptic curve. ($0.12T_{Mul}$)

T_{INV} is time complexity of performing an inverse operation. ($0.073T_{Mul}$)

T_{HASH} is time complexity of performing a hash function. (Negligible)

We have compared our scheme's computational cost with the schemes in [13, 15, 16, and 18]. You can see the results in Table 1. Ming et al. scheme [15] and Liu et al. scheme [13] are not secure [18]. Zhang et al. scheme [16] has no security proof. Li et al. scheme [18] is secure and has less computational cost compared to [13, 16]. Comparing our scheme with mentioned schemes in Table 1 shows that our scheme has much lower computational cost.

7 CONCLUSION

CBS schemes use traditional public key infrastructures and identity-based signatures advantages and have no certificate management problem in PKI and key escrow in IBS. In this paper, a new CBS scheme based on elliptic curve cryptography is proposed. The security of our scheme is proven under the ECDL assumption and in the random oracle model. Comparing our scheme with existing pairing-free CBS schemes shows that ours has less computational cost.

7 REFERENCES

[1] A. Shamir, Identity-based cryptosystems and signature schemes, in: G.R. Blakely, D. Chaum (Eds.), CRYPTO 1984, vol. 196, LNCS, 1985, pp. 47–53.

[2] C. Gentry, Certificate-based encryption and the certificate revocation problem, in: E. Biham (Ed.), EUROCRYPT 2003, LNCS, vol. 2656, 2003, pp. 272–293.

[3] S.S. Al-Riyami, K.G. Paterson, Certificateless public key cryptography, in: Laih, C.S. (Ed.), ASIACRYPT 2003, LNCS, vol. 2894, 2003, pp. 452–473.

[4] J.G. Li, X.Y. Huang, Y. Mu, W. Susilo, Q.H. Wu, Certificate-based signature: security model and efficient construction, in: J. Lopez, P. Samarati, J.L. Ferrer (Eds.), EuroPKI 2007, LNCS, vol. 4582, 2007, pp. 110–125.

[5] B.G. Kang, J.H. Park, S.G. Hahn, A certificate-based signature scheme, in: T. Okamato (Ed.), CT-RSA, 2004, LNCS, vol. 2964, 2004, pp. 99–111.

[6] M.H. Au, J.K. Liu, W. Susilo, T.H. Yuen, Certificate based (linkable) ring signature, in: E. Dawson, D.S. Wong (Eds.), ISPEC 2007, LNCS, vol. 4464, 2007, pp. 79–92.

[7] L.H. Wang, J. Shao, Z.F. Cao Pandu Rangan, M. Mambo, A. Yamamura, A certificate-based proxy cryptosystem with revocable proxy decryption power, in: K. Srinathan, C., M. Yung (Eds.), INDOCRYPT 2007, LNCS, vol. 4859, 2007, pp. 297–311.

[8] W. Wu, Y. Mu, W. Susilo, X.Y. Huang, Certificate-based signatures: new definitions and a generic construction from certificateless signatures, in: K.I. Chung, K. Sohn, M. Yung (Eds.), WISA 2008, LNCS, vol. 5379, 2009, pp. 99–114.

[9] J.G. Li, L.Z. Xu, Y.C. Zhang, Provably secure certificate-based proxy signature schemes, Journal of Computers 4 (6) (2009) 444–452.

[10] J.G. Li, X.Y. Huang, Y. Mu, W. Susilo, Q.H. Wu, Constructions of certificate-based signature secure against key replacement attacks, Journal of Computer Security 18 (3) (2010) 421–449.

[11] N. Koblitz, A. Menezes, S.A. Vanstone, The state of elliptic curve cryptography, Designs, Codes and Cryptography 9 (2/3) (2000) 173–193.

[12] L. Chen, Z. Chen, N. Smart. Identity-based key agreement schemes from pairings. Int J Inform Secure 2007; 6:213–41.

[13] J.K. Liu, J. Baek, W. Susilo, J. Zhou, Certificate-based signature scheme without pairings or random oracles, in: T.C. Wu et al. (Eds.), ISC 2008, LNCS, vol. 5222, 2008, pp. 285–297.

[14] J. Zhang, On the security of a certificate-based signature scheme and its improvement with pairings, in: F. Bao, H. Li, G. Wang (Eds.), ISPEC 2009, LNCS, vol. 5451, 2009, pp. 47–58.

[15] Y. Ming, Y. Wang, Efficient certificate-based signature scheme, IAS 2009, vol.2, IEEE, 2009, pp. 87–90.

[16] J. Zhang, H. Chen, Q. Geng, An efficient certificate-based signature scheme without pairings, in: WCSE 2009, IEEE, vol.2, 2009, pp. 44–48.

[17] J.G. Li, X.Y. Huang, Y.C. Zhang, L.Z. Xu, An efficient short certificate-based signature scheme, Journal of Systems and Software 85 (2) (2012) 314–322.

[18] Li, J., Wang, Z., & Zhang, Y. Provably secure certificate-based signature scheme without pairings. Information Sciences, 2013, 233, 313-320.

[19] D. Pointcheval, J. Stern, Security proofs for signature schemes, in: EURPCRYPT 1996, LNCS, vol. 1070, 1996, pp. 387–398.

[20] Y.F. Chung, K.H. Huang, F. Lai, T.S. Chen, ID-based digital signature scheme on the elliptic curve cryptosystem, Computer Standards and Interfaces 29 (6) (2007) 601–604.