An Efficient Certificateless Signature Scheme

Wun-She Yap[1], Swee-Huay Heng[2] and Bok-Min Goi[1] *

[1] Centre for Cryptography and Information Security, FOE
Multimedia University, 63100 Cyberjaya, Malaysia
{wsyap,bmgoi}@mmu.edu.my
[2] Centre for Cryptography and Information Security, FIST
Multimedia University, Jln Ayer Keroh Lama, 75450 Melaka, Malaysia
shheng@mmu.edu.my

Abstract. Certificateless public key cryptography (CLPKC) is a paradigm to solve the inherent key escrow problem suffered by identity-based cryptography (IBC). While certificateless signature is one of the most important security primitives in CLPKC, there are relatively few proposed schemes in the literature. In this paper, we manage to construct an efficient certificateless signature scheme based on the intractability of the computational Diffie-Hellman problem. By using a shorter public key, two pairing computations can be saved in the verification algorithm. Besides, no pairing computation is needed in the signing algorithm. The proposed scheme is existential unforgeable in the random oracle model. We also present an extended construction whose trust level is the same as that of a traditional signature scheme.

Keywords: Certificateless, signature scheme, bilinear pairing

1 Introduction

The concept of identity-based cryptography (IBC) was formulated by Shamir in 1984 [15] to achieve implicit certification. In IBC, each user has his own identity (ID). ID is used as a certified public key, thus certificate can be omitted in authenticating the public key. However, since all private keys of the users are generated by a trusted third party (TTP) called private key generator (PKG), *private key escrow problem* is inherent in the system. To solve the inherent key escrow problem in IBC, a new paradigm called certificateless public key cryptography (CLPKC) was introduced by Al-Riyami and Paterson [1] in 2003.

Many certificateless public key encryption (CLPKE) schemes [1, 13, 6, 2, 4, 16] have been proposed since CLPKC was introduced whereas there are relatively few certificateless signature (CLS) schemes [1, 12, 11] in the literature. The current trend in e-commence has increased the dependence of both organization and individual on the sensitive information stored and communicated electronically using the computer systems. This has spurred a need to guarantee the confidentiality, authenticity and integrity of data and user. Thus we see

* The authors acknowledge the Malaysia IRPA grant (04-99-01-00003-EAR)

the importance of proposing CLS scheme to guarantee the authenticity without using certificate.

The first CLS scheme was proposed by Al-Riyami and Paterson in [1] but there is no security proof provided. Besides, their CLS scheme has been proven insecure in their defined model by Huang et al. [11]. They showed an attack that can successfully forge a certificateless signature by replacing the public key of the signer. They also fixed the scheme in the same paper. Unfortunately, the fixed scheme required more pairing computations as compared to the original scheme proposed in [1]. Yum and Lee proposed a generic construction of CLS based on an identity-based signature (IBS) scheme and a traditional public key signature scheme in [17] which is a different approach in constructing CLS. The merit of the above approach is that the resulting CLS scheme can achieve the same trust level as that of a traditional signature scheme. Li, Chen and Sun [12] proposed another CLS based on bilinear pairing by referring to the work of Cha and Cheon [5]. In [12], the signing algorithm of the proposed scheme is very simple and does not involve any pairing computation, but the verification algorithm requires four expensive pairing computations. This inspires us to come out with a more efficient CLS scheme which requires lesser bilinear pairing computations.

Our Contributions. We outline some results we achieve below.

1. EFFICIENCY. Our proposed scheme is more efficient than those schemes proposed in [1, 12, 11] because lesser bilinear pairing computations are required. Besides, our public key length is shorter.
2. SECURITY. We provide a detailed security proof based on the computational Diffie-Hellman assumption. Schemes [12] and [11] did not provide the complete security proofs while scheme [1] did not provide any security proof.
3. TRUST LEVEL. Our extended construction achieves the same trust level as that of a traditional signature scheme as was proposed in [17], which is better than those schemes proposed in [12, 11].

Organization. The remainder of the paper is organized as follows. In Section 2, we introduce some preliminaries which will be referred later. In Section 3, we review the definition, the security model and the attack model of CLS. In Section 4, we propose our CLS. In Section 5, we present its security analysis. In Section 6, we present an extended construction which achieves trust level 3. Finally, we conclude this paper in Section 7.

2 Preliminaries

In this section, we present some mathematical problems which help in realizing CLS. Bilinear pairing is an important primitive for many cryptographic CLPKE schemes [1, 13, 6, 2, 4, 16] and CLS schemes [1, 12, 11]. We describe some of its key properties below.

Notation: Throughout this paper, $(G_1, +)$ and $(G_2, \cdot)$ denote two cyclic groups of prime order q. A *bilinear map*, $e : G_1 \times G_1 \rightarrow G_2$ satisfies the following properties:

1. Bilinearity: For all $P, Q, R \in G_1$, $e(P+Q, R) = e(P, R)e(Q, R)$ and $e(P, Q+R) = e(P, Q)e(P, R)$.
2. Non-degeneracy: $e(P, Q) \neq 1$.
3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G_1$.

Note that a *bilinear map* is symmetric such that, $e(aP, bP) = e(bP, aP) = e(P, P)^{ab}$ for $a, b \in Z_q^*$.

Definition 1. *Computational Diffie-Hellman Problem (CDHP): The CDHP in (G_1, G_2, e) is such that given (P, aP, bP) with uniformly random choices of $a, b \in Z_q^*$, find abP. The CDH assumption states that there is no polynomial time algorithm with a non-negligible advantage in solving the CDHP.*

Our security proofs will yield reductions to the CDHP in groups generated by generator $\mathcal{LG}$. To make statements about the security of our scheme, we will assume that there is no polynomial time algorithm with a non-negligible advantage in solving the CDHP in groups generated by $\mathcal{LG}$.

3 Certificateless Signature Scheme

3.1 Definition

We now review the definition of a CLS [1].

A certificateless signature scheme is a digital signature scheme comprised of the following seven algorithms:

1. *Setup* is a probabilistic algorithm that takes security parameter k as input and returns the system parameters, *params* and *master-key*.
2. *Partial-Private-Key-Extract* is a deterministic algorithm that takes *params*, *master-key* and an identifier for entity A $ID_A \in \{0, 1\}^*$ as inputs. It returns a partial private key D_A.
3. *Set-Secret-Value* is a probabilistic algorithm that takes as input *params* and outputs a secret value x_A.
4. *Set-Private-Key* is a deterministic algorithm that takes *params*, D_A and x_A as inputs. The algorithm returns a (full) signing key S_A.
5. *Set-Public-Key* is a deterministic algorithm that takes *params* and x_A as inputs and outputs a public key P_A.
6. *Sign* is a probabilistic algorithm that accepts a message $m \in M$, a user identity ID_A, *params* and S_A to produce a signature σ.
7. *Verify* is a deterministic algorithm that takes a signature σ, message m, *params*, ID_A and P_A as inputs and outputs *true* if the signature is correct, or $\perp$ otherwise.

Definition 2. *We say that a certificateless signature scheme is correct if the following condition holds.*

If $\sigma = Sign(m,\ ID_A,\ S_A,\ params)$ *and* $S = (\sigma,\ m,\ ID_A,\ P_A,\ params)$, *then* $Verify(S) = true$.

3.2 Adversarial Model

As defined in [1], there are two types of adversaries with different capabilities. In CLS, we assume Type I Adversary, $\mathcal{A}_{\mathcal{I}}$ acts as a malicious key generator centre (KGC) while Type II Adversary, $\mathcal{A}_{\mathcal{II}}$ acts as a dishonest user.

CLS Type I Adversary: Adversary $\mathcal{A}_{\mathcal{I}}$ does not have access to *master-key*, but $\mathcal{A}_{\mathcal{I}}$ may replace public keys.
CLS Type II Adversary: Adversary $\mathcal{A}_{\mathcal{II}}$ does have access to *master-key*, but cannot replace public keys of entities.

Both types of adversary may request public keys, extract partial private and private keys and make sign queries. Here are several natural restrictions on both types of adversary:

1. $\mathcal{A}$ cannot extract the private key for the challenge identity ID_{ch} at any point.
2. During the attack, $\mathcal{A}$ cannot make a sign query on the forged message, m for the combination of identity (ID_{ch}, P_{ch}).

Besides, $\mathcal{A}_{\mathcal{I}}$ cannot both replace the public key for ID_{ch} and extract the partial private key for ID_{ch}. Similarly, $\mathcal{A}_{\mathcal{I}}$ cannot request the partial private key for any identity if the corresponding public key has already been replaced.

The standard notion of security for signature scheme is the security against existential forgery on adaptive chosen message attacks [10]. The formal security model was presented neither in [1] nor [12]. We follow the one defined in [11] here.

A CLS is secure against existential forgery on adaptive chosen message and ID attacks against adversary, $\mathcal{A}$ if no polynomial time algorithm $\mathcal{A}$ has a non-negligible advantage against a challenger $\mathcal{C}$ in the following game:

`Setup`: The challenger, $\mathcal{C}$ takes a security parameter k and runs the *Setup* algorithm. It gives $\mathcal{A}$ the resulting system parameters *params*. If $\mathcal{A}$ is of Type I, then the challenger keeps *master-key* to itself, else it gives *master-key* to $\mathcal{A}$.

`Attack`: $\mathcal{A}$ issues a sequence of requests, each request being either a partial private key extraction, a private key extraction, a request for a public key, a replace public key command or a sign query for a particular entity. These queries may be asked adaptively, but are subjected to the rules on adversary behaviors defined above.

`Forgery`: Finally, $\mathcal{A}$ outputs a signature σ on message m signed by a user who holds ID_{ch} and public key P_{ch}. The only restriction is that (m, ID_{ch}, P_{ch}) does not appear in the set of previous sign queries. $\mathcal{A}$ wins the game if Verify$(\sigma, m, ID_{ch}, P_{ch})$ is *true*. The advantage of $\mathcal{A}$ is defined as the probability that it wins.

4 Proposed CLS Scheme

In this section, we show how to combine the techniques used in [1, 5, 6, 16] with the elegancy of bilinear pairing to construct an efficient CLS scheme. Our $verify$ algorithm requires two pairing computations only while four and five pairing computations are required in [12] and [11] respectively. Besides, messages signing is fast since it involves no pairing computation.

The proposed CLS scheme is constructed by the following seven algorithms:

1. *Setup*: Given a security parameter k, the algorithm works as follows:
   (a) Run $\mathcal{LG}$ to output descriptions of groups G_1 and G_2 of prime order q and a pairing $e : G_1 \times G_1 \rightarrow G_2$.
   (b) Choose an arbitrary generator $P \in G_1$.
   (c) Select a random $s \in Z_q^*$, and set $P_0 = sP$.
   (d) Choose a cryptographic hash function $H_1 : \{0,1\}^* \rightarrow G_1$ and $H_2 : \{0,1\}^* \times G_1 \rightarrow Z_q^*$.

   The system parameters are $params = \langle G_1, G_2, e, q, P, P_0, H_1, H_2 \rangle$. The message space is $M = \{0,1\}^*$. The *master-key* is $s \in Z_q^*$.

2. *Set-Partial-Private-Key*: Given *params* and *master-key*, this algorithm works as follows: Compute $Q_A = H_1(ID_A) \in G_1$ and output a partial private key, $D_A = sQ_A \in G_1$.

3. *Set-Secret-Value*: Given *params*, select a random value $x_A \in Z_q^*$ where x_A is the secret value.

4. *Set-Private-Key*: Set private key $S_A = (x_A Q_A + D_A)$.

5. *Set-Public-Key*: Given *params* and the *secret value* x_A, this algorithm computes $P_A = x_A P \in G_1$.

6. *Sign*: Given *params*, ID_A, message m and private key S_A, the algorithm works as follows:
   (a) Compute $Q_A = H_1(ID_A) \in G_1$.
   (b) Choose a random value, $r \in Z_q^*$ and set $U = rQ_A \in G_1$.
   (c) Set $h = H_2(m||U) \in Z_q^*$.
   (d) Compute $V = (r + h)S_A$.
   (e) Set $\sigma = (U, V)$ as the signature of m.

7. *Verify*: Given signature σ, ID_A, m and P_A, this algorithm works as follows:
   (a) Compute $Q_A = H_1(ID_A) \in G_1$.
   (b) Compute $h = H_2(m||U) \in Z_q^*$.
   (c) Check whether $\langle P, P_0 + P_A, U + hQ_A, V\rangle$ is a valid Diffie-Hellman tuple, i.e. by verifying whether $e(P, V) = e(P_0 + P_A, U + hQ_A)$. If not, then reject the signature else accept it.

In a CLS, *Setup* and *Partial-Private-Key-Extract* are performed by a KGC. A partial private key D_A is given to a user A by the KGC through a secure channel. CLS can solve the inherent key escrow problem which suffered by IBS since *Set-Secret-Value*, *Set-Private-Key* and *Set-Public-Key* are executed by the user A itself. In order to generate a signature, the user A needs to run *Sign* algorithm with input S_A, m, ID_A and *params*. Finally, the receiver can verify A's signature by running *Verify* algorithm with input σ, m, ID_A and *params*. It is clear that the user identifier ID used in the *Sign* algorithm will provide the non repudiation of signature.

5 Analysis of the Proposed Scheme

In this section, we analyze the correctness, the performance and the existential unforgeability of our proposed scheme.

5.1 Correctness

The correctness of the proposed scheme can be easily verified with the following:

$$\begin{aligned} e(P, V) &= e(P, (r + h)(D_A + x_A Q_A) \\ &= e(P, (r + h)(sQ_A + x_A Q_A) \\ &= e((s + x_A)P, (r + h)Q_A \\ &= e(P_0 + P_A, U + hQ_A) \end{aligned}$$

5.2 Performance

There are three major cost operations in constructing cryptographic schemes, namely, Pairing (p), Scalar Multiplication (s) and Exponentiation (e). The pairing operations are expensive compared with scalar multiplication and exponentiation. Table 5.2 shows the comparison of the existing CLS schemes and our proposed scheme in terms of the public key length and efficiency of *Sign* and *Verify* algorithms (we do not consider the pre-computation here). We can see that our scheme is the most efficient scheme in terms of the number of pairing operations required and the length of public key.

Table 1. Comparison of the CLS Schemes

Schemes	AP2003[1]	LCS2005[12]	HMSZ2005[11]	Proposed scheme
Sign	1p+3s	2s	2p+3s	2s
Verify	4p+1e	4p+2s	5p+1e	2p+3s
Public Key Length	2 points	2 points	2 points	1 point

5.3 Security

We now present the security analysis of our proposed scheme. The proof of Theorem 1 is provided. We prove the security in the random oracle model [3].

Theorem 1. *The proposed CLS scheme is existential unforgeable against the $\mathcal{A}_{\mathcal{I}}$ adversary in the random oracle model under the CDH assumption in G_1.*

Proof. (Theorem 1) Let $\mathcal{B}$ be a CDH attacker. Suppose that $\mathcal{B}$ is given an instance (q, P, aP, bP). Let $\mathcal{A}_{\mathcal{I}}$ be a forger that breaks the proposed signature scheme under adaptive chosen message attack. We show how $\mathcal{B}$ can use $\mathcal{A}_{\mathcal{I}}$ to solve the CDH problem, that is to compute abP. First, $\mathcal{B}$ sets $P_0 = aP$ where P_0 denotes the KGC's public key. $\mathcal{B}$ then gives (q, P, P_0) to $\mathcal{A}_{\mathcal{I}}$.

Next, $\mathcal{B}$ randomly selects an index I such that $1 \leq I \leq q_{H_1}$, where q_{H_1} denotes the maximum number of queries to the random oracle H_1. $\mathcal{B}$ also sets $P_I = xP$ where x is selected at random from Z_q^*. Let P_I serve as user I's original public key. Adversary $\mathcal{B}$ then works by interacting with $\mathcal{A}_{\mathcal{I}}$ in a chosen message attack game as follows:
H_1 *queries*: $\mathcal{B}$ maintains a list of tuples $\langle ID_i, Q_i, y_i, x_i, P_i \rangle$ which is denoted as H_1^{list}. The list is initially empty, and when $\mathcal{A}_{\mathcal{I}}$ queries H_1 on input $ID_i \in \{0,1\}^*$, $\mathcal{B}$ responds as follows:

1. If ID_i has appeared on the H_1^{list}, then $\mathcal{B}$ responds with $H_1(ID_i) = Q_i \in G_1$.
2. If ID_i has not appeared on the list and ID_i is the I-th distinct H_1 query made by $\mathcal{A}_{\mathcal{I}}$, then $\mathcal{B}$ outputs $H_1(ID_I) = Q_I = bP$ and adds the entry $\langle ID_I, Q_I, \perp, x, P_I \rangle$ to the H_1^{list} where $Q_I = bP$ and $P_I = xP$. Else, $\mathcal{B}$ picks $y_i, x_i \in Z_q^*$ at random and responds with $H_1(ID_i) = Q_i = y_i P \in G_1$. $\mathcal{B}$ then adds $\langle ID_i, Q_i, y_i, x_i, P_i \rangle$ to the H_1^{list} where $Q_i = y_i P$ and $P_i = x_i P$.

Notice that with this specification of H_1, the partial private key for ID_i with $i \neq I$ is equal to $y_i P_0$ while the public key for ID_i with $i \neq I$ is $P_i = x_i P$, and the private key for ID_i with $i \neq I$ is $x_i Q_i + y_i P_0$. These can all be computed by $\mathcal{B}$.
H_2 *queries*: When $\mathcal{A}_{\mathcal{I}}$ issues a query on (m_i, U_i) to H_2, $\mathcal{B}$ picks a random $h_i \in Z_q^*$ and returns it as answer.
Attack: Now $\mathcal{A}_{\mathcal{I}}$ launches *Phase 1* of its attack by making a series of queries, each of which is either a *Partial Private Key Extraction*, a *Private Key Extraction*, a *Request for Public Key*, a *Replace Public Key* or a *Sign Queries*. $\mathcal{B}$ replies to these queries as follows:

Partial Private Key Extraction: Suppose the query is on ID_i with $i \neq I$, then $\mathcal{B}$ replies with $D_i = y_i P_0$ (notice that $D_i = y_i P_0 = a y_i P = a H_1(ID_i)$). Else if $i = I$, $\mathcal{B}$ aborts.
Private Key Extraction: Suppose the query is on ID_i. We can assume that the public key for ID_i has not been replaced. If $i \neq I$, then $\mathcal{B}$ replies with $x_i Q_i + D_i$. Else if $i = I$, $\mathcal{B}$ aborts.
Request for Public Key: If the query is on ID_i with $i \neq I$, then $\mathcal{B}$ replies with $P_i = x_i P$ by accessing the H_1^{list}. Else if $i = I$, $\mathcal{B}$ replies with P_I.
Replace Public Key: Suppose the query is to replace the public key for ID_i with value P_i'. If $i \neq I$, then $\mathcal{B}$ replaces P_i with P_i' in the H_1^{list} and updates the tuple to $\langle ID_i, Q_i, y_i, x_i', P_i' \rangle$. Else if $i = I$, then $\mathcal{B}$ replaces P_i with P_i' in the H_1^{list} and updates the tuple to $\langle ID_I, Q_I, \perp, x_I', P_I' \rangle$.
Sign Queries: Note that at any time during the simulation, equipped with those private keys and partial private keys for any $ID_i \neq ID_I$, $\mathcal{A}_{\mathcal{I}}$ is able to generate signatures on any message. For $ID_i = ID_I$, assume that $\mathcal{A}_{\mathcal{I}}$ issues a query (m_i, P_I) where m_i denotes a message and P_I denotes a current public key chosen by $\mathcal{A}_{\mathcal{I}}$ to the signing oracle whose private key is associated with ID_I. Upon receiving this, $\mathcal{B}$ creates a signature as follows:

1. Select $h_i, z_i \in Z_q^*$ at random.
2. Compute $U_i = z_i P - h_i Q_I$ where $Q_I = H_1(ID_I) = bP$.
3. Compute $V_i = z_i(P_0 + P_I)$.
4. Set $h_i = H_2(m_i || U_i)$.
5. Return (U_i, V_i) as a signature on m_i.

It is straightforward to verify that *Private Key Extraction* and *Sign* produce *valid* private keys and signatures respectively. From the above simulation of *Partial Private Key Extraction*, H_1 and H_2, it can be easily seen that the distribution of the simulated outputs are identical to those in the real attack.
Forgery: The next step of the simulation is to apply the forking technique formalized in [14]. Let $(m, (U, V), ID, P_I)$ be a forgery that output by $\mathcal{A}_{\mathcal{I}}$ at the end of the attack. If $\mathcal{A}_{\mathcal{I}}$ does not output $ID = ID_I$ as a part of the forgery then $\mathcal{B}$ aborts (the probability that $\mathcal{B}$ does not abort the simulation is $O(1/q_{H_1})$). $\mathcal{B}$ then replays $\mathcal{A}_{\mathcal{I}}$ with the same random tape but different choice of the hash function H_2 to get another forgery $(m, (U, V'), ID, P_I)$. Notice that the hash values $h \neq h'$ on (m, U) for the two choice of H_2. Now a standard argument for the outputs of the forking lemma can be applied as follows: since both are valid signatures, $\langle P, P_0 + P_I, U + hQ_I, V \rangle$ and $\langle P, P_0 + P_I, U + h'Q_I, V' \rangle$ are valid Diffie-Hellman tuples. More precisely, we have $V = (x + a)(U + hQ_I)$ and $V' = (x + a)(U + h'Q_I)$. $\mathcal{B}$ consequently obtains the following:

$$\begin{aligned} V - V' &= (x + a)(hQ_I - h'Q_I) \\ &= x(h - h')Q_I + a(h - h')bP \\ &= x(h - h')Q_I + (h - h')abP \end{aligned}$$

Thus, it is not difficult to see that

$$abP = \{(V - V') - x(h - h')Q_I)\} \cdot (h - h')^{-1}.$$

This completes our proof. □

Theorem 2. *The proposed CLS scheme is existential unforgeable against the $\mathcal{A}_{\mathcal{II}}$ adversary in the random oracle model under the CDH assumption in G_1.*

Due to page limitation, the proof of Theorem 2 will be presented in the full version of the paper.

6 Extended Construction

The public key cryptosystem can be classified into three trust levels referred to the trust assumption of the TTP as defined by Girault [9]. In order to extend our signature scheme to achieve trust level 3, we use the binding technique which ensures that users can only create one public key for which they know the corresponding private key. This technique was first employed in [1] in order to prevent the KGC from issuing two valid partial private keys for a single user.

First, user A must fix its secret value, x_A and its public key, $P_A = x_A P$. Then, KGC generates the partial private key D_A for user A by returning sQ_A where $Q_A = H_1(ID_A||P_A)$. However, a drawback of the binding technique is that the user A can no longer choose another secret value x'_A to generate a new public key P'_A since the partial private key D_A remains the same. This technique has also been used in [17]. Now, the KGC who replaces user's public key will be implicated in the event of dispute: the existence of two working public keys for an identity can only result from the existence of two partial private keys binding that identity to two different public keys. Thus, only the KGC could have created these two partial private keys since only the KGC has access to the *master-key*.

To adopt this binding technique in our scheme, we should execute *Set-Secret-Value* and *Set-Public-Key* before executing *Set-Partial-Private-Key*. This extended version is identical to our proposed CLS scheme above except the differences in the sequence of the execution of the algorithms and that the value $Q_A = H_1(ID_A||P_A)$ will be used instead.

7 Conclusion

We have presented a more efficient CLS scheme compared with other existing CLS schemes. Our scheme is provably secure in the random oracle model under the CDH assumption. To the best of our knowledge, this scheme has the shortest public key length. By adopting the techniques used in [1, 5, 6, 16], two pairing computations used in authenticating public key can be saved. Besides, we also managed to extend our CLS to achieve trust level 3 by adopting the technique used in [1]. Some future research includes finding a provably secure CLS scheme in the standard model and extending the CLS to ring signature scheme [7] and concurrent signature scheme [8].

References

1. S.S. Al-Riyami and K.G. Paterson. Certificateless Public Key Cryptography. *In Proceedings of ASIACRYPT 2003*, LNCS 2894, pp. 452-473, Springer-Verlag, 2003.
2. S.S. Al-Riyami and K.G. Paterson. CBE from CL-PKE: A Generic Construction and Efficient Schemes. *In Proceedings of PKC 2005*, LNCS 3386, pp. 398-415, Springer-Verlag, 2005.
3. M. Bellare and P. Rogaway. Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. *In Proceedings of CCCS 1993*, pp. 62-73, ACM press, 1993.
4. J. Baek, R. Safavi-Naini and W. Susilo. Certificateless Public Key Encryption Without Pairing. *In Proceedings of ISC 2005*, LNCS 3650, pp. 134-148, Springer-Verlag, 2005.
5. J. Cha and J. Cheon. An Idendity-Based Signature from Gap Diffie-Hellman Groups. *In Proceedings of PKC 2003*, LNCS 2567, pp. 18-30, Springer-Verlag, 2003.
6. Z.H. Cheng and R. Comley. Efficient Certificateless Public Key Encryption. *Cryptology ePrint Archive*, Report 2005/012, 2005. http://eprint.iacr.org/2005/012.
7. S.S.M. Chow, L.C.K. Hui and S.M. Yiu. Idendity Based Threshold Ring Signature. *In Proceedings of ICISC 2004*, LNCS 3506, pp. 218-232, Springer-Verlag, 2005.
8. L. Chen, C. Kudla and K.G. Paterson. Concurrent Signatures. *In Proceedings of EUROCRYPT 2004*, LNCS 3027, pp. 287-305, Springer-Verlag, 2004.
9. M. Girault. Self-Certified Public Keys. *In Proceedings of EUROCRYPT 1991*, LNCS 547, pp. 490-497, Springer-Verlag, 1991.
10. S. Goldwasser, S. Micali and R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. *SIAM Journal of Computing*, vol. 17, no. 2, pp. 281–308, 1988.
11. X. Huang, W. Susilo, Y. Mu and F. Zhang. On the Security of Certificateless Signature Schemes from Asiacrypt 2003. *In Proceedings of CANS 2005*, LNCS 3810, pp. 13-25, Springer-Verlag, 2005.
12. X. Li, K. Chen and L. Sun. Certificateless Signature and Proxy Signature Schemes from Bilinear Pairings. *Lithuanian Mathematical Journal*, Vol 45, pp. 76-83, Springer-Verlag, 2005.
13. Y.R. Lee and H.S. Lee. An Authenticated Certificateless Public Key Encryption Scheme. *Cryptology ePrint Archive*, Report 2004/150, 2004. http://eprint.iacr.org/2004/150.
14. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. *In Proceedings of EUROCRYPT 1996*, LNCS 1070, pp. 387-398, Springer-Verlag, 1996.
15. A. Shamir. Identity Based Cryptosystems and Signature Scheme. *In Proceedings of CRYPTO 1984*, LNCS 196, pp. 47-53, Springer-Verlag, 1984.
16. Y. Shi and J. Li. Provable Efficient Certificateless Public Key Encrytion. *Cryptology ePrint Archive*, Report 2005/287. http://eprint.iacr.org/2005/287.
17. D.H. Yum and P.J. Lee. Generic Construction of Certificateless Signature. *In Proceedings of ACISP 2004*, LNCS 3108, pp. 200-211, Springer-Verlag, 2004.