An Efficient Certificateless Signature Scheme

Rafael Castro and Ricardo Dahab

UNICAMP, Brazil,
rafael.castro@gmail.com,rdahab@ic.unicamp.br

Abstract. In this paper we present a certificateless signature (CLS) scheme secure in the Random Oracle Model. This scheme requires no pairing computations for signature generation and only two for signature verification. As far as we know, this is the only CLS scheme to require less than four pairing computations on signature verification.
Key Words: Certificateless Public-Key Cryptography

1 Introduction

Certificateless Public Key Cryptography (CL-PKC) is a novel approach to PKC proposed in [ARP03] that aims at solving the problem of key escrow, inherent to ID-Based Public Key Cryptography (ID-PKC) [Sha85]. The main feature of ID-PKC is the ability to use the user's identity as his public key, solving, in theory, the problems involved in managing Public Key Infrastructures (PKI). However, some sort of secret information must be involved in the computation of private keys, creating the need for a trusted entity, the Trust Authority (TA), which is responsible for computing users' private keys based on their identity. The fact that the TA has access to users' private keys, and can thus impersonate them, is a form of *key escrow*, and is the main barrier in ID-PKC's wide adoption. Certificateless Cryptography tries to change this by introducing a compromise between the complete absence of public keys and a full-blown PKI. In CL-PKC each user has a public key created by themselves using some secret random information. But to sign on behalf of this public key, under a certain identity, secret information supplied by a Key Generation Center *(KGC)* must be used together with the random information used to generate the public key. So, in principle, no certificates need to be checked because only a user with the correct ID could have obtained the secret information from the KGC.

Our main contribution in this paper is a certificateless signature *(CLS)* scheme that only requires 2 pairing computations on the signature verification procedure, and none in the singing procedure. It is, as far as our knowledge goes, the most efficient CLS scheme available.

1.1 Organization

In Section 2 we review a few important concepts and present some useful definitions. In Section 3 we discuss the security of certificateless signatures. In Section 4 we present our signature scheme and in Section 5 we analyze its security. Finally, in Section 6 we give a quick comparison with other CLS schemes.

2 Preliminaries

2.1 Bilinear maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be groups such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ that satisfies the following properties:

1. **Bilinearity.** For all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $e(aP, bQ) = e(P, Q)^{ab}$;
2. **Non-degeneracy.** Let Q be a generator of $\mathbb{G}_2$ and $\psi()$ an homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. Then $e(\psi(Q), Q) \neq 1$.

Additionally, we want the map e to be efficiently computable. Such a bilinear map is called *admissible*. In the particular case where $\mathbb{G}_1 = \mathbb{G}_2$, the map is called *symmetric*. Examples of bilinear maps widely used in cryptography are the Weil pairing (as in [BF01]) and the Tate pairing.

2.2 Security assumptions and hard problems

We base our security reductions on a few important definitions presented below.

Definition 1. ***Decision Diffie-Hellman Problem*** **(DDHP).** *Given a multiplicative group* $(\mathbb{G}, .)$*, and elements* $\alpha, \alpha^a, \alpha^b, \alpha^c \in \mathbb{G}$*, decide wether* $c = ab$*.*

Definition 2. ***Computational Diffie-Hellman Problem*** **(CDHP).** *Given a multiplicative group* $(\mathbb{G}, .)$*, and elements* $\alpha, \alpha^a, \alpha^b \in \mathbb{G}$*, compute* $X = \alpha^{ab}$*.*

Definition 3. ***Generalized Computational Diffie-Hellman Problem*** **(GCDHP).** *Given a multiplicative group* $(\mathbb{G}, .)$*, and elements* $\alpha, \alpha^a, \alpha^b \in \mathbb{G}$*, compute* (α^{abc}, α^c)*.*

The GCDHP bears a relation to the Generalized Bilinear Diffie-Hellman Problem used by Al-Riyami & Paterson [ARP03], similar to the one between the CDHP and the Bilinear Diffie-Hellman Problem [BF01]: it is a strictly weaker security assumption.

2.3 Certificateless signatures

Definition 4. *A Certificateless Signature* (CLS) *scheme consists of five polynomial-time algorithms:*

- **Setup.** Run by the KGC to initialize the system. Receives a security parameter 1^k and returns the public and private master keys (mpk, msk).
- **Extract Partial Private Key.** Takes as input (mpk, msk) and the identity $ID \in \{0, 1\}^*$, and outputs the private partial key D_{ID}, which is assumed to be given to the correct user through a secure channel.
- **Generate Key Pair.** Takes as input mpk and generates the user's public key P_{ID} and corresponding private key S_{ID}.
- **CL-Sign.** Takes as input mpk, the user's identity ID, the pair of keys (D_{ID}, S_{ID}), and a message M. Outputs a signature σ on M.
- **CL-Verify.** Takes as input mpk, ID, P_{ID}, M and the signature σ, and outputs `ACCEPT` if and only if σ is a valid signature by user U_{ID} on M under public key P_{ID}.

3 Security of Certificateless Signatures

Since the work of Goldwasser, Micali and Rivest [GMR88] the standard security notion for a signature scheme is *existential unforgeabillity against adaptively chosen message attacks* (EU-CMA). In CLS, EU-CMA security is traditionally expressed by two similar games: in both cases an attacker $\mathcal{A}$ is trying to break the EU-CMA security of the scheme and runs as follows:

1. The challenger $\mathcal{C}$ generates a master key pair *(mpk,msk)* whose distribution is indistinguishable from that obtained from running *(mpk,msk)* = $\texttt{Setup}(1^k)$.
2. The attacker $\mathcal{A}$ runs on input *mpk* and (possibly) some extra information *aux*. During its execution $\mathcal{A}$ has access to some oracles, which are described subsequently. If $\mathcal{A}$ does not abort, it should output a forgery (U^*, M^*, γ^*).

The attacker wins the game if $\texttt{CL-Verify}(\mathbb{U}^*,\mathbb{M}^*,\gamma^*)$ = `ACCEPT` and (U^*,M^*) has not been queried to the signing oracle. A CLS scheme is secure if it can be proven that any polynomial-time attacker has negligible chance of breaking the scheme.

Definition 5. ***Negligible Function.*** *A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if, for each polynomial p, there exists an integer $N(p)$ such that $|f(x)| \leq \frac{1}{|p(x)|}$ for all $x \geq N(p)$.*

Now we have to define the oracles adversaries may have access to.

- **RevealPublicKey.** The adversary supplies an identity ID_i and the challenger returns the corresponding public key P_{ID_i}. If no such key exists, a new one is generated.
- **RevealPartialKey.** The adversary supplies an identity ID_i and the challenger returns the corresponding partial private key D_{ID_i}. If no such key exists, a new one is generated.
- **RevealSecretValue.** The adversary supplies an identity ID_i and the challenger returns the secret value corresponding to its public key. If there is no public key associated with ID_i, one is created. If the public key has been replaced and the corresponding secret value is unknown, the challenger returns $\perp$.
- **ReplacePublicKey.** The adversary supplies an identity ID_i and a public key $P^*_{ID_i}$. The challenger sets the public key corresponding to ID_i to $P^*_{ID_i}$.
- **Sign.** The adversary supplies an identity ID_i, a message M and optionally a secret value x_{ID_i}; if x_{ID_i} is not supplied, then set x_{ID_i} to the current value of S_{ID_i}. The challenger returns a signature of M as generated by $\texttt{CL-Sign}(ID_i, x_{ID_i}, D_{ID_i}, M)$. Notice that if an invalid secret value is supplied, an invalid signature will be generated.

If the oracle is required to generate signatures under public keys that were replaced by the adversary (as in [ARP03]), it is called a **StrongSign** oracle. If this requirement is dropped, then we have a **WeakSign** oracle.

The security of certificateless signatures is thus expressed by two similar games, respectively against $\mathcal{A}_I$ and $\mathcal{A}_{II}$, defined as follows:

Game I: *Let $\mathcal{C}_I$ be the challenger algorithm and k be a security parameter:*

1. *$\mathcal{C}_I$ executes* `Setup`(1^k) *and obtains (mpk, msk);*
2. *$\mathcal{C}_I$ runs $\mathcal{A}_I$ on 1^k and mpk. During its run, $\mathcal{A}_I$ has access to the following oracles:* `RevealPublicKey, RevealPartialKey, RevealSecretValue, ReplacePublicKey, QueryHash, Sign;`
3. *$\mathcal{A}_I$ outputs (ID^*, M^*, σ^*).*

$\mathcal{A}_I$ wins the game if `CL-Verify`(`params`,ID^*,P_{ID^*},M^*,σ^*)=`ACCEPT` and both conditions below hold:

- `Sign`(ID^*,M^*) was never queried;
- `RevealPartialKey`(ID^*) was also never queried.

Game II: *Let $\mathcal{C}_{II}$ be the challenger algorithm and k be a security parameter:*

1. *$\mathcal{C}_{II}$ executes* `Setup`(1^k) *and obtains (mpk, msk);*
2. *$\mathcal{C}_{II}$ runs $\mathcal{A}_{II}$ on 1^k and (mpk, msk). During its run, $\mathcal{A}_{II}$ has access to the following oracles:* `RevealPublicKey, RevealPartialKey, RevealSecretValue, ReplacePublicKey, QueryHash, Sign;`
3. *$\mathcal{A}_{II}$ outputs (ID^*, M^*, σ^*).*

$\mathcal{A}_{II}$ wins the game if `CL-Verify`(`params`,ID^*,P_{ID^*},M^*,σ^*)=`ACCEPT` and all conditions below hold:

- `Sign`(ID^*,M^*) was never queried;
- `RevealSecretValue`(ID^*) was never queried;
- `ReplacePublicKey`(ID^*, .) was never queried.

For a longer discussion on CLS security models, we refer the reader to [HWZD07].

4 A secure and efficient CLS scheme

In this section we present the main contribution of this paper. This scheme is closely related to the IBS from [YCK04], being an adaptation of the former to the certificateless setting.

- **Setup.** Given a GDH group $\mathbb{G}$ of order p, with an admissible pairing e and its generator P, pick $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$ (a random element from $\mathbb{Z}_p^*$) and set $P_{pub} = sP$. Choose two hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_2 : \{0,1\}^* \times \mathbb{G}^2 \rightarrow \mathbb{Z}_p^*$. The public information is $mpk \overset{def}{=} (P, P_{pub}, H_1, H_2)$, and the master secret is $msk = s$.
- **Extract Partial Private Key.** Given an identity ID, compute $Q_{ID} = H_1(ID)$ and $D_{ID} = sH_1(ID)$. Output D_{ID} as the partial private key corresponding to $Q_{ID} = H_1(ID)$.
- **Generate Key Pair.** Generate a secret $x \overset{R}{\leftarrow} \mathbb{Z}_p$. The public key is $P_{ID} = xP_{pub}$. The user's private key is $S_{ID} = x$.

– **Sign.** Given the user's private keys (D_{ID}, x) and a message M, pick a number $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and output a signature $\sigma = (V, R)$ where $R = rP$ and $V = rQ_{ID} + hxD_{ID}$, where $h = H_2(M, R, P_{ID})$.
– **Verification.** Given a signature $\sigma = (V, R)$ of a message M for an identity ID, compute $h = H_2(M, R, P_{ID})$. The signature is accepted if and only if

$$e(P, V) = e(Q_{ID}, R + hP_{ID}).$$

4.1 Correctness

The correctness of our signature scheme can be easily verified as follows:

$$\begin{aligned} e(P, V) &= e(P, rQ_{ID} + hxD_{ID}) \\ &= e(P, rQ_{ID} + hxsQ_{ID}) \\ &= e((r + hxs)P, Q_{ID}) \\ &= e(Q_{ID}, R + hP_{ID}). \end{aligned}$$

5 Security analysis of our scheme

In this section we discuss our scheme's security under the security model discussed in Section 3. The main theorem concerning the security of our scheme is:

Theorem 1. *If the CDHP and the GCDHP are hard in $\mathbb{G}$, then our CL-PKS scheme is secure under the Random Oracle Model.*

This result follows directly from Lemmas 1 and 2 below.

Lemma 1. *Let $\mathcal{A}_1$ be a Type-I adversary that (T_1, ϵ_1)-breaks our scheme with at most q_S signature queries and q_{H_2} identity-hash queries. Assume that $\epsilon_1 \geq (10(q_S + 1)(q_S + q_{H_2}))/2^k$. Then the GCDHP can be solved within running time $T_1' \leq (120686 q_{H_2} T_1)/(\epsilon_1(1 - \frac{1}{2^k}))$, where k is a security parameter.*

Proof. We are given an instance (P, aP, bP) of the GCDHP. We construct the challenger $\mathcal{C}_1$ that will use $\mathcal{A}_1$ to solve the GCDHP. On the setup phase, $\mathcal{C}_1$ sets P as the generator of the group, and sets $P_{pub} \leftarrow aP$. $\mathcal{C}_1$ then randomly chooses the target user identity ID^*, gives $(ID^*, \texttt{params})$ to $\mathcal{A}_1$ and starts to answer oracle queries with the following procedures:

– **ID-Hash Query(ID_i).** If $ID_i = ID^*$ then let $Q_i = bP$ and $y_i = \perp$. Else, $\mathcal{C}_1$ generates a random y_i and lets $Q_i = y_i P$. In any case, $\mathcal{C}_1$ makes $P_i = x_i = \perp$, saves the tuple $(ID_i, Q_i, P_i, y_i, x_i)$ and returns $H_1(ID_i) = Q_i$.
– **Partial Key Extraction(ID_i).** Find the tuple $(ID_i, Q_i, P_i, y_i, x_i)$. If it does not exist, or $y_i = \perp$, then $\mathcal{C}_1$ aborts. Otherwise answer with $D_{ID_i} = y_i P_{pub} = y_i(aP)$. Note that $\mathcal{A}_1$ is not allowed to request the partial key for ID^*.

– **Secret Value Extraction(ID_i).** Find the tuple $(ID_i, Q_i, P_i, y_i, x_i)$. If it does not exist, or $y_i = \perp$, then $\mathcal{C}_1$ aborts. If $x_i = \perp$, choose a random $x_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and save its value. In any case, return x_i.
– **Public Key Extraction(ID_i).** Find the tuple $(ID_i, Q_i, P_i, y_i, x_i)$. If it does not exist, or $y_i = \perp$, then $\mathcal{C}_1$ aborts. If $x_i = \perp$, execute `Secret Value Extraction` to generate a private key. Answer with $P_{ID_i} = x_i P_{pub}$.
– **Public Key Replacement(ID_i,P_i').** Find the tuple (ID_i, Q_i, P_i, x_i). If it does not exist, $\mathcal{C}_1$ aborts. Otherwise $\mathcal{C}_1$ sets $x_i = \perp$ and $P_i = P_i'$.
– **Message-Hash Query(M_j,R_j,P_{ID_j}).** If $H_2(M_j, R_j, P_{ID_j})$ is not defined, $\mathcal{C}_1$ chooses a random value $h_j \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and stores $H_2(M_j, R_j, P_{ID_j}) = h_j$. In any case, $H_2(M_j, R_j, P_{ID_j})$ is returned.
– **Sign Query(ID_i,M_j).** When $\mathcal{A}_1$ asks for a signature of user ID_i on message M_j, $\mathcal{C}_1$ does the following:
  1. Find the tuple $(ID_i, Q_i, P_i, y_i, x_i)$, aborting if it cannot be found;
  2. choose $r_t, h_j \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$;
  3. compute $R_t = r_t - h_j P_i$ and $V_t = r_t(x_i P)$;
  4. if $H_2(M_j, R_t, P_{ID_i})$ is defined, abort; else, set $H_2(M_j, R_t, P_{ID_i}) = h_j$;
  5. return the signature $\sigma = (V_t, R_t)$.

If the challenger $\mathcal{C}_1$ does not abort, $\mathcal{A}_1$ will output a valid forgery $\gamma_1 = (ID, m, h, R, V)$ with probability ϵ_1. The probability of $\mathcal{C}_1$ aborting is $\frac{1}{2^k}$. We can now use the replay technique from [PS00] to obtain another forgery $\gamma_2 = (ID, m, h', R', V')$ such that $R = R'$, but $h \neq h'$ within time $T_1' \leq \frac{120686 q_{H_1} T_1}{\epsilon_1(1-\frac{1}{2^k})}$. Now, let Y be computed as follows:

$$Y = \frac{V - V'}{h - h'} = \frac{(hxD_{ID^*} - h'xD_{ID^*})}{h - h'} = xD_{ID^*} = xabP.$$

So $(Y, P_{ID}) = (xabP, xP)$ is an answer to our GCDHP instance. □

The proof for Type-2 adversaries is very similar, and is outlined below.

Lemma 2. *Let $\mathcal{A}_2$ be a Type-2 adversary that (T_2, ϵ_2)-breaks our scheme with at most q_S signature queries and q_{H_2} identity-hash queries. Assume that $\epsilon_2 \geq (10(q_S + 1)(q_S + q_{H_2}))/2^k$. Then, the CDHP can be solved within running time $T_2' \leq (120686 q_{H_2} T_2)/(\epsilon_2(1 - \frac{1}{2^k}))$, where k is a security parameter.*

Proof. We are given an instance (P, aP, bP) of the CDHP. We construct the challenger $\mathcal{C}_2$ that will use $\mathcal{A}_2$ to solve the CDHP. On the setup phase, $\mathcal{C}_2$ sets P as the generator of the group, randomly chooses $s \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sets $P_{pub} = sP$. $\mathcal{C}_2$ then randomly chooses the target user identity ID^* and gives (ID^*, `params`) to $\mathcal{A}_2$ and starts to answer oracle queries with the following procedures:

– **ID-Hash Query(ID_i).** If $ID_i = ID^*$ then make $Q_i = aP$ and $y_i = \perp$. Else, $\mathcal{C}_2$ generates a random y_i and makes $Q_i = y_i P$. In any case, $\mathcal{C}_2$ makes $P_i = x_i = \perp$, saves the tuple $(ID_i, Q_i, P_i, y_i, x_i)$ and returns $H_1(ID_i) = Q_i$.

- **Partial Key Extraction(ID_i).** Find the tuple $(ID_i, Q_i, P_i, y_i, x_i)$. If it does not exist, or $y_i = \perp$, then $\mathcal{C}_2$ aborts. Otherwise answer with $D_{ID_i} = y_i P_{pub} = y_i(aP)$.
- **Secret Value Extraction(ID_i).** Find the tuple $(ID_i, Q_i, P_i, y_i, x_i)$. If it does not exist, or $y_i = \perp$, then $\mathcal{C}_2$ aborts. If $x_i = \perp$, choose a random $x_i \xleftarrow{R} \mathbb{Z}_p$ and save it. Return x_i.
- **Public Key Extraction(ID_i).** Find the tuple $(ID_i, Q_i, P_i, y_i, x_i)$. If it does not exist, or $y_i = \perp$, then $\mathcal{C}_2$ aborts. If $ID_i = ID^*$, then return $P_{ID_i} = bP$. If $x_i = \perp$, execute `Private Key Extraction` to generate a private key. In any case, return $P_{ID_i} = y_i P_{pub}$.
- **Message-Hash Query(M_j,R_j,P_{ID_j}).** If $H_2(M_j, R_j, P_{ID_j})$ is not defined, $\mathcal{C}_2$ chooses a random value $h_j \xleftarrow{R} \mathbb{Z}_p$ and stores $H_2(M_j, R_j, P_{ID_j}) = h_j$. In any case, $H_2(M_j, R_j, P_{ID_j})$ is returned.
- **Sign Query(ID_i,M_j).** When $\mathcal{A}_2$ asks for a signature by user ID_i on message M_j, $\mathcal{C}_2$ does the following:
  1. Find the tuple $(ID_i, Q_i, P_i, y_i, x_i)$, aborting if it cannot be found;
  2. choose random $r_t, h_j \xleftarrow{R} \mathbb{Z}_p^*$;
  3. compute $R_t = r_t - h_j P_i$ and $V_t = r_t(x_i P)$;
  4. if $H_2(M_j, R_t, P_{ID_i})$ is already defined, abort; else, set $H_2(M_j, R_t, P_{ID_i}) = h_j$;
  5. return the signature $\sigma = (V_t, R_t)$.

If the challenger $\mathcal{C}_2$ does not abort, $\mathcal{A}_2$ will output a valid forgery $\gamma = (ID, m, h, R, V)$ with probability ϵ_2. The probability of $\mathcal{C}_2$ aborting is $\frac{1}{2^k}$. $\mathcal{C}_2$ then replays $\mathcal{A}_2$ and, by the same replay technique as in Lemma 1, two valid forgeries, γ and γ', are obtained within time $T_2' \leq \frac{120686 q_{H_1} T_2}{\epsilon_2(1-\frac{1}{2^k})}$.

Now let Y be computed as follows:

$$Y = \frac{V - V'}{h - h'} = \frac{(hbD_{ID^*} - h'bD_{ID^*})}{h - h'} = D_{ID^*} = sabP.$$

So $s^{-1}Y = abP$ is the answer to our CDHP instance.

□

5.1 On Malicious KGC Attacks

It has recently come to our attention that our scheme is vulnerable to a new attack outlined in [ACL+06]. In this attack, the KGC maliciously generates the system parameters so that it can impersonate users (a specific user or every user of the system, depending on the severity of the attack). Our scheme is vulnerable to a weaker version of the attack, in which the KGC can choose only *one* identity to attack, as outlined below:

Assume the KGC wants to attack user U^*, whose identity is ID^*. Now it can compute $Q_{ID^*} = H_1(ID^*)$, choose $k \xleftarrow{R} \mathbb{Z}_p^*$, set the main generator $P \leftarrow k^{-1}Q_{ID^*}$ and proceed with the rest of the setup normally. Now the KGC knows that

$kP = Q_{ID^*}$, so signatures can be forged for any public key without knowledge of the full private key because

$$V = rQ_{ID^*} + xhD_{ID^*} = rQ_{ID^*} + khP_{ID^*}$$

This attack is not covered by the security model presented here and involves the malicious generation of system parameters by the KGC. Almost every other CL-PKS scheme available is also vulnerable to some form of this attack (with the notable exception of the generic scheme from [HWZD06]), but there is a simple solution to this weak version of the attack: use the binding technique proposed in [ARP03].

There, the authors propose that the KGC issue partial keys on the identity appended with the user's public key ($Q_{ID_i} = H_1(ID_i||P_{ID_i})$), instead of the identity alone. This solves various security concerns of CL-PKC, but in our specific attack scenario the KGC has no way of generating the system parameters maliciously, since it has no way of knowing what the user's public key will be beforehand. Thus it cannot mount the attack.

6 Conclusion

In this paper we presented a new certificateless signature *(CLS)* scheme whose security was proved under the Random Oracle Model. It is, as far as we know, the only CLS scheme that requires less than 4 pairing computations on the signature verification step, making it the most efficient alternative available, as shown in Table 1.

Scheme	Signing Cost	Verification Cost
Al-Riyami & Paterson, as in [HSMZ05]	2	5
Li, Chen & Sun [LCS05]	0	4
Zhang et al. [ZWXF06]	0	4
Liu, Au & Susilo [LAS06][1]	0	6
Castro & Dahab	0	2

Table 1. The cost columns indicate the number of pairing computations.

7 Acknowledgments

We would like thank Augusto Jun Devegili for many insightful comments on preliminary versions of this paper.

[1] This is the only CLS scheme proven secure in the Standard Model, so it is expected to be less efficient than the others.

References

[ACL+06] Man Ho Au, Jing Chen, Joseph K. Liu, Yi Mu, Duncan S. Wong, and Guomin Yang. Malicious kgc attacks in certificateless cryptography. Cryptology ePrint Archive, Report 2006/255, 2006. http://eprint.iacr.org/.

[ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[HSMZ05] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. On the security of certificateless signature schemes from asiacrypt 2003. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *CANS*, volume 3810 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 2005.

[HWZD06] Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, and Xiaotie Deng. Key replacement attack against a generic construction of certificateless signature. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP*, volume 4058 of *Lecture Notes in Computer Science*, pages 235–246. Springer, 2006.

[HWZD07] Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, and Xiaotie Deng. Certificateless signature: a new security model and an improved generic construction. *Des. Codes Cryptography*, 42(2):109–126, 2007.

[LAS06] Joseph K. Liu, Man Ho Au, and Willy Susilo. Self-generated-certificate public key cryptography and certificateless signature / encryption scheme in the standard model. Cryptology ePrint Archive, Report 2006/373, 2006. http://eprint.iacr.org/.

[LCS05] X. Li, K. Chen, and L. Sun. Certificateless signature and proxy signature schemes from bilinear pairings. *Lithuanian Mathematical Journal*, 45(1):76–83, 2005.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(3):361–396, 2000.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[YCK04] HyoJin Yoon, Jung Hee Cheon, and Yongdae Kim. Batch verifications with id-based signatures. In Choonsik Park and Seongtaek Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 233–248. Springer, 2004.

[ZWXF06] Zhenfeng Zhang, Duncan S. Wong, Jing Xu, and Dengguo Feng. Certificateless public-key signature: Security model and efficient construction. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 293–308, 2006.