

# A Certificateless Signature Scheme based on Bilinear Pairing Functions

Routo Terada \*

Denise H. Goya<sup>†</sup>

**Abstract**— We construct a Certificateless Public Key Signature scheme - CL-PKS, i.e., a cryptographic signature scheme which does not require any Digital Certificate to verify a signature generated by a private key, based on asymmetric bilinear pairing functions. Our scheme does not allow the so-called key escrow. We analyze both its efficiency and security: it is more efficient than previously published CL-PKS schemes, with shorter signatures and public keys; we prove it is strong against adaptively chosen message attacks, based on the computational difficulty of the Diffie-Hellman Problems.

## 1. Introduction

The concept of Identity Based Encryption – IBE – system was proposed by [13] for which the public key can be the user’s identity itself, such as the user’s e-mail address.

[5] presented an IBE system based on bilinear pairing functions, that requires a Public Key Generator – PKG. The PKG needs to be trusted in the sense that it can generate any of the private keys, i.e., it can exercise the so-called *key escrow*, which is undesirable in many applications. On the other hand this system does not require the so-called Public Key Infrastructure – PKI – with its complex and costly management of Digital Certificates. Recently [3] proposed an Identity Based Signature - IBS - scheme based on *asymmetric* bilinear pairing functions.

[1] proposed a Certificateless Public Key Encryption – CL-PKE – scheme, i.e., a cryptographic scheme which does not require either a Digital Certificate to certify the public key or a PKI. It is also based on bilinear pairing functions. In CL-PKE there is a trusted third party - TTP - who owns a **master-key**, but it is used only partially to generate a user’s private key so that this key cannot be generated by the TTP alone, which is possible in IBE. In [1] the TTP is called Key Generator Center - KGC - and the notion of CL-PKE is formalized, and a security analysis is presented.

Informally, the KGC uses its **master-key** and publicly known system parameters to build a *partial* private key  $D_A$  associated to a user’s identity  $ID_A$ ; as in IBE, this private key  $D_A$  must be delivered securely to the user. Then this user composes  $D_A$  with another random secret data to generate its own private key  $t_A$  and *publishes* the corresponding public key  $N_A$ ; hence, the KGC does *not* have access to the user’s private key. The public key  $N_A$  is bound to the identity  $ID_A$  and

thus it does not need to be authenticated. Due to this last fact, it is relevant to consider an adversary may replace the public key  $N_A$  by a fake one [1]. But since the KGC knows the **master-key** and the partial private key  $D_A$ , it is able to mount a public key replacement attack, so in the CL-PKE security analysis a necessary assumption is that the KGC is trusted not to do such replacement. This trust is equivalent to the trust in a Certificate Authority of a PKI. As in [1], we consider two types of adversaries: (1) Type I Adversary: it does not have access to the KGC’s **master-key** but can replace any public key  $N_A$  (corresponding to the private key  $t_A$ ) of its choice. (2) Type II adversary: it does have access to the **master-key** but cannot replace any public key  $N_A$ .

Besides CL-PKE, a certificateless signature scheme - CL-PKS - is presented in [1], but there is no security analysis. In the case of message signing, the public key  $N_A$  must be sent with the message. Recently [10] showed this CL-PKS is insecure, and a fix is proposed based on *symmetric* bilinear pairing functions, as defined in Section 2.

### Our contribution

We construct a CL-PKS scheme which: (1) is based on *asymmetric* bilinear pairing functions; (2) does not allow key escrow by the KGC; and does not require Digital Certificates; (3) is more efficient on computation than previously published CL-PKS ([1], [2], [10], [3], [17]); (4) and is secure in the sense that no adversary may forge a signature unless the *q-Strong Diffie-Hellman Problem* (*q-SDH*), and the *Bilinear Pairing Inversion Problem* (*BPI*), defined in Section 2, are polynomial time solvable.

Since our CL-PKS is based on *asymmetric* bilinear pairing functions  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  (as opposed to the symmetric pairing in [10] and [17]), ordinary elliptic curves such as MNT curves [11] or BN curves [4] should be used as long as a publicly computable but not necessarily invertible isomorphism  $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$  is available.

\* rt@ime.usp.br, University of Sao Paulo, supported by Japan Soc. for the Promotion of Science Fellowship S06207

<sup>†</sup> University of Sao Paulo

Preliminary concepts are defined in Section 2. Our CL-PKS scheme is in Section 3. In Section 4 we analyze the complexity. Security analyses are in Sections 5. Conclusions are in Section 6.

## 2. Preliminaries

### 2.1. Bilinear maps

Let  $k$  be a security parameter and  $\mathbb{G}_1$  be an additive group of prime order  $p > 2^k$  and  $\mathbb{G}_2$  and  $\mathbb{G}_T$  be multiplicative groups of the same order. Let  $P$  and  $Q$  be generators of  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , respectively, and let  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  (possibly  $\mathbb{G}_1 = \mathbb{G}_2$ ) be a mapping with the following properties: (1) Bilinearity:  $e()$  is bilinear iff  $\forall (R, S) \in \mathbb{G}_1 \times \mathbb{G}_2, \forall a, b \in F_p : e(aR, bS) = e(R, S)^{ab}$ , where  $F_p$  is a finite field of order  $p$ ; (2) Non-degeneracy:  $e()$  is non-degenerate if  $\forall R \in \mathbb{G}_1 : e(R, S) = 1_{\mathbb{G}_T} \forall S \in \mathbb{G}_2$  iff  $R = \mathcal{O}$ , the identity element of  $\mathbb{G}_1$ ; (3) Efficient computability:  $e()$  is efficiently computable if  $\forall (R, S) \in \mathbb{G}_1 \times \mathbb{G}_2 : e(R, S)$  can be computed in polynomial-time; (4) There exists an efficient and publicly computable (not necessarily invertible) homomorphism  $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$  such that  $\psi(Q) = P$ .

When  $\mathbb{G}_1 \neq \mathbb{G}_2$   $e()$  is said to be *asymmetric* and ordinary elliptic curves such as MNT curves [11] or BN curves [4] can be used and the trace map can be used as the isomorphism  $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$  if  $\mathbb{G}_2$  is properly chosen [14]. When  $\mathbb{G}_1 = \mathbb{G}_2$   $e()$  is said to be *symmetric* and supersingular curves can be used and  $\psi()$  is the identity.

For a group  $\mathbb{G}$  of prime order, we denote  $\mathbb{G}^* = \mathbb{G} \setminus \mathcal{O}$ , where  $\mathcal{O}$  is the identity element of  $\mathbb{G}$ .

### 2.2. Computational problems

For security analysis a computationally hard problem was formalized in [6] and is recalled in the following.

**Definition 1.** ([6]) Let  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$  be groups of prime order  $p$  and let  $P$  and  $Q$  be generators of  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , respectively. The q-Strong Diffie-Hellman Problem (q-SDH) in the groups  $\mathbb{G}_1, \mathbb{G}_2$  takes as input a  $(q+2)$ -tuple  $(P, Q, \alpha Q, \alpha^2 Q, \dots, \alpha^q Q)$  and aims to find a pair  $(c, \frac{1}{c+\alpha} P)$  with  $c \in Z_p^*$ .

A second computationally hard problem is defined next. Its computational hardness is proved in [16].

**Definition 2.** ([16]) Let  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$  be groups of prime order  $p$  and let  $P$  and  $Q$  be generators of  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , respectively. Let  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  with the properties described before. The Bilinear Pairing Inversion Problem (BPI) takes as input  $(Q \in \mathbb{G}_2, e(P, Q) \in \mathbb{G}_T)$  and aims to compute  $P \in \mathbb{G}_1$ .

### 2.3. Formal model of Certificateless Public Key Signature - CL-PKS

The formal structure we use for Certificateless Public Key Signature consists of seven algorithms as follows.

**Definition 3.** ([1][10]) A CL-PKS (Certificateless Public Key Signature) scheme consists of the following polynomial-time algorithms:

(1) Setup: This is a probabilistic algorithm. It takes as input a security parameter  $1^k$  and returns **params**, a list of system parameters, and a **master-key**. (2) Partial-Private-Key-Extract: This is a deterministic algorithm that accepts as input  $ID_A$ , the identity for the user  $A$ , **params** and **master-key**, to produce the partial private key  $D_A$ . (3) Set-Secret-Value: This probabilistic algorithm outputs a secret information  $t_A$  for  $ID_A$ . (4) Set-Private-Key: This is a deterministic algorithm that accepts as input  $ID_A$ , the partial private key  $D_A$ , the secret information  $t_A$  and **params**, to produce the private signing key  $(D_A, t_A)$ . (5) Set-Public-Key: This is a deterministic algorithm that accepts as input  $ID_A$ , the secret information  $t_A$  and **params**, to produce the public key  $N_A$ . (6) CL-Sign: This is a probabilistic algorithm. It takes as input  $ID_A$ , the private signing key  $(D_A, t_A)$ , **params** and a message  $M$ , to produce a signature  $\sigma$ . (7) CL-Verify: This deterministic algorithm takes as input  $ID_A$ , the public key  $N_A$ , **params**, a message  $M$  and a signature  $\sigma$ . It returns a bit  $b$ ;  $b = 1$  means that the signature is accepted, whereas  $b = 0$  means rejected.

## 3. Proposed CL-PKS scheme

Our concrete CL-PKS scheme is defined as the following set of algorithms.

**Setup** Given a security parameter  $k \in Z$  the Key Generation Center - KGC: (1) Generates two cyclic groups  $\mathbb{G}_1, \mathbb{G}_2$ ,  $\mathbb{G}_1 \neq \mathbb{G}_2$ , and  $\mathbb{G}_T$  of prime order  $p > 2^k$  and a bilinear pairing  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ . Chooses randomly two generators  $P \in \mathbb{G}_1^*, Q \in \mathbb{G}_2^*$  such that  $P = \psi(Q)$  where  $\psi()$  is an isomorphism (not necessarily invertible) between  $\mathbb{G}_1^*$  and  $\mathbb{G}_2^*$ . (2) Computes  $g = e(P, Q) \in \mathbb{G}_T$ , which is public. (3) Chooses randomly the master-key  $s \in Z_p^*$  and compute  $Q_{pub} = sQ$ . (4) Chooses two hash functions:  $H_1 : \{0, 1\}^* \rightarrow Z_p^*$ ,  $H_2 : \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{G}_T \times \mathbb{G}_T \rightarrow Z_p^*$ .

The message/plaintext space is  $\mathcal{M} = \{0, 1\}^*$ . The signature space is  $\mathcal{S} = \mathbb{G}_1 \times Z_p^*$ . The system secret **master-key** is  $s$ . The system public parameters are

$$\text{params} = \langle p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(), \psi, P, Q, Q_{pub}, g, H_1, H_2 \rangle$$

**Partial-Private-Key-Extract** Given an identifier  $ID_A \in \{0, 1\}^*$ , **params** and the **master-key**  $s$  the KGC: (1) computes  $D_A = \frac{1}{H_1(ID_A) + s} P \in \mathbb{G}_1^*$ . (2) gives the partial secret key  $D_A$  securely to  $A$ , the owner of  $ID_A$ .

**Set-Secret-Value** Given **params**, the entity  $A$  selects at random a secret information  $t_A \in Z_p^*$  for identity  $ID_A$ .

**Set-Private-Key** Given  $ID_A$ ,  $D_A$ ,  $t_A$ , and **params**, the entity  $A$  keeps safely  $A$ 's signing private key pair  $(D_A, t_A) \in \mathbb{G}_1^* \times Z_p^*$ .

**Set-Public-Key** Given  $ID_A$ ,  $t_A$ , and **params**, the entity  $A$  computes its public key  $N_A = g^{t_A} \in \mathbb{G}_T$ .  $A$  publishes  $N_A$ .

**CL-Sign** Given  $M \in \mathcal{M} = \{0, 1\}^*$ , an identity  $ID_A$ , **params** and the published public key  $N_A = g^{t_A}$  and  $A$ 's secret key pair  $(D_A, t_A)$ ,  $A$  signs  $M$  as follows: (1) chooses randomly  $x \in Z_p^*$ . (2) computes:  $r :=$

$g^x \in \mathbb{G}_T$ ,  $h := H_2(M, ID_A, N_A, r) \in Z_p^*$ ,  $S := (x + ht_A)D_A \in \mathbb{G}_1$ . (3) a signature on  $M$  is  $\sigma := (S, h) \in \mathbb{G}_1 \times Z_p^*$ .

**CL-Verify** Given plaintext  $M$ , and  $\sigma = (S, h)$ , the signature, it outputs 1 (i.e., accepts  $\sigma$  as authentic) if and only if:  $h = H_2(M, ID_A, N_A, r')$  where  $r' := e(S, H_1(ID_A)Q + Q_{pub})(N_A)^{-h}$ . Otherwise it outputs 0.

### 3.1. Correctness of the proposed CL-Sign

To prove the CL-Sign is correct, it is enough to remember the pairing  $e(\cdot)$  is bilinear, and to prove  $r' = r$ , as shown next:

$$\begin{aligned}
r' &= e(S, H_1(ID_A)Q + Q_{pub})(N_A)^{-h} \\
&= e([(x + ht_A)D_A], \\
&\quad H_1(ID_A)Q + Q_{pub})(N_A)^{-h} \\
&= e([(x + ht_A)\frac{1}{H_1(ID_A)+s}P], \\
&\quad H_1(ID_A)Q + sQ)(N_A)^{-h} \\
&= e([(x + ht_A)\frac{1}{H_1(ID_A)+s}P], \\
&\quad [H_1(ID_A) + s]Q)(N_A)^{-h} \\
&= e([(x + ht_A)P], Q)^{\frac{1}{H_1(ID_A)+s}(H_1(ID_A)+s)} \\
&\quad (N_A)^{-h} \\
&= e([(x + ht_A)P], Q)(N_A)^{-h} \\
&= e(P, Q)^{(x+ht_A)}(g^{t_A})^{-h} \\
&= g^{(x+ht_A)}g^{-ht_A} \\
&= g^x = r
\end{aligned}$$

## 4. Complexity of the proposed CL-PKS

[1] presents a CL-PKS scheme without security analysis. Recently [10] showed this CL-PKS is insecure, and a fix is proposed based on *symmetric* bilinear pairing functions, i.e., the particular case of  $\mathbb{G}_1 = \mathbb{G}_2$ . A summary of [10] is:  $U = vtD + rP$ ,  $v = H(M, R, e(tD, P))$ , and asks  $v = ? H(M, e(U, P)e(Q_i, -N)^v, e(Q_i, N))$ , where  $D, t$  are secret,  $N$  is public,  $R = e(rP, P)$ ,  $Q_i = H_1(ID)$ . Our CL-PKS scheme is based on more general *asymmetric* bilinear pairing functions (i.e., the general case of  $\mathbb{G}_1 \neq \mathbb{G}_2$ ). [17] presents a CL-PKS scheme which is also based on *symmetric* bilinear pairing functions. A summary of [17] is:  $U = rP, V = D + rH_2(M, ID, P, U) + th_3$ , and asks  $e(V, P) = ? e(h_1, P_{pub})e(h_2, U)e(h_3, N)$ .

Recall that  $\mathbb{G}_1$  is an additive group of prime order  $p > 2^k$  and  $\mathbb{G}_2$  and  $\mathbb{G}_T$  are multiplicative groups of the same order;  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ . In some of the previous papers  $\mathbb{G}_1 = \mathbb{G}_2$ , an additive group. But in this paper  $\mathbb{G}_1 \neq \mathbb{G}_2$ . Let the basic operations be denoted as follows: (1) P: bilinear pairing  $e(R, S) \in \mathbb{G}_T, (R, S) \in \mathbb{G}_1 \times \mathbb{G}_2$  (2) E: exponentiation  $g^x \in \mathbb{G}_T, g \in \mathbb{G}_T, x \in Z_p^*$  (3) S: scalar multiplication in additive group  $xD \in \mathbb{G}_1, x \in Z_p^*, D \in \mathbb{G}_1^*$  (4) M: multiplication of points in multiplicative group  $QR \in \mathbb{G}_2^*, Q \in \mathbb{G}_2^*, R \in \mathbb{G}_2^*$  (5) A: addition of points in additive groups  $(Q + R) \in \mathbb{G}_1, Q \in \mathbb{G}_1, R \in \mathbb{G}_1$  (6) H: hash computation  $H_j(\cdot) \in Z_p^*$

Then we have the following table of comparison with the number of computations in each of the published similar schemes of CL-PKS based on bilinear pairing. This table shows that our CL-PKS is more efficient in the number of bilinear pairing operations, which is the

relatively most expensive ones: zero in the CL-Sign, and one in the CL-Verify (as opposed to 4 in previous papers).

Signing a message	CL-Sign					
CL-PKS	P	S	M	E	A	H
[1]	1	3	0	0	1	1
[10]	2	2	0	0	1	1
[17]	0	3	0	0	2	2
Our CL-PKS	0	1	0	1	0	1

  

Verifying a signature	CL-Verify					
CL-PKS	P	S	M	E	A	H
[1]	4	0	1	1	0	1
[10]	4	0	1	1	0	2
[17]	4	0	2	0	0	3
Our CL-PKS	1	1	1	1	1	2

In the next table we see that a signature in our CL-PKS is shorter than in Zhang et al. [17], since an integer in  $Z_p^*$  is shorter than a point in  $\mathbb{G}_1$ . For a given length  $n$  a proper choice of the security parameter  $k$  is possible so that our signature is shorter than in [1] and [10]. Compressed pairings may be used as in [15]. Huang et al. [10] and Zhang et al. [17] use symmetric pairings; if they used asymmetric pairings as we do, their public keys must be in  $\mathbb{G}_2$ . If MNT curves [11] are used together with compressed pairings, our CL-PKS would have public keys as large as in [10] and [17]; if BN curves [4] are used, our public keys can be shorter.

CL-PKS	Signature Space
[1]	$\mathbb{G}_1 \times \{0, 1\}^n$
[10]	$\mathbb{G}_1 \times \{0, 1\}^n$
[17]	$\mathbb{G}_1 \times \mathbb{G}_1$
Our CL-PKS	$\mathbb{G}_1 \times Z_p^*$

  

CL-PKS	Public Key Space
[1]	$\mathbb{G}_1$
[10]	$\mathbb{G}_1$
[17]	$\mathbb{G}_1$
Our CL-PKS	$\mathbb{G}_T$

## 5. Security Proof of Our CL-PKS

We follow the security model as defined by Al-Riyami and Patterson in [1]; it is detailed in the Appendix.

In this model, the Type-I adversary against CL-PKS does not know the **master-key** but may replace any user's public key. It plays the Game I against a challenger as described below. The Type-II adversary against CL-PKS knows the **master-key** but cannot replace any user's public key. It plays Game II against a challenger as described below. We will prove both adversaries are Existentially Unforgeable against Adaptive Chosen Message Attackers (EUF-CMA).

Informally if we let  $\mathcal{A}^I$  and  $\mathcal{A}^{II}$  denote Type-I and Type-II attackers, respectively, playing Game-I and Game-II (as defined below), CL-PKS scheme is said to be EUF-CMA of a CL-PKS if the success probability of both  $\mathcal{A}^I$  and  $\mathcal{A}^{II}$  are negligible (see Definition 4 in the Appendix).

To prove the security against an adversary  $\mathcal{A}^I$ , we will use a variant of the Game-I: the challenger gives

an identity  $ID$  and the adversary should be able to produce a valid signature for that  $ID$  and some message  $M$ ; we say a CL-PKS scheme is existentially unforgeable against adaptive chosen message and *given identity* Type-I attacks if any probabilistic polynomial time (on  $k$ ) adversary has a negligible probability in gaining success. The Lemma 1 below constructs an attacker  $\mathcal{A}$ , challenged on a *given identity*, if our CL-PKS scheme is under a Type-I attacker. Then in Lemma 2 we prove that if there is such a polynomial time (on  $k$ ) adversary  $\mathcal{A}$  then there is a polynomial time algorithm to solve the  $q$ -SDH Problem, which is a hard problem (see Section 2.2).

**Lemma 1.** *If there is an algorithm  $\mathcal{A}^I$  for an adaptively chosen message Type-I attack against our CL-PKS scheme with running time  $t$ , advantage  $\epsilon$  and making  $q_{h_1}$  queries to random oracle  $H_1$ , then there is an algorithm  $\mathcal{A}$  for an adaptively chosen message and given identity Type-I attack which has advantage  $\epsilon_1 \geq \epsilon(1 - \frac{1}{2^k})/q_{h_1}$ , with a running time  $t_1 \leq t$ . Moreover,  $\mathcal{A}$  asks the same number of queries as  $\mathcal{A}^I$  does.*

*Proof.* This proof is similar to the proof of Lemma 1 in [8]. Without loss of generality, we may assume that for any  $ID$ ,  $\mathcal{A}^I$  queries  $H_1(ID)$ , Extract Private Key( $ID$ ), Extract Partial Private Key( $ID$ ), Request Public Key( $ID$ ), and Replace Public Key( $ID$ ) at most once. The steps done by  $\mathcal{A}$  are:

(1) Choose independently and randomly  $r \in \{1, \dots, q_{h_1}\}$ . Denote by  $ID_i$  the input for the  $i$ -th query to  $H_1$  asked by  $\mathcal{A}^I$ . Let  $ID'_i$  be  $ID$  if  $i = r$ , and  $ID_i$  otherwise. Define  $H'_1(ID_i) = H_1(ID'_i)$ , Extract Private Key'( $ID_i$ ) = Extract Private Key( $ID'_i$ ), Extract Partial Private Key'( $ID_i$ ) = Extract Partial Private Key( $ID'_i$ ), Request Public Key'( $ID_i$ ) = Request Public Key( $ID'_i$ ), Replace Public Key'( $ID_i$ ) = Replace Public Key( $ID'_i$ ) and Sign'( $ID_i, M$ ) = Sign( $ID'_i, M$ ).

(2) Run  $\mathcal{A}^I$  with the given system parameters.  $\mathcal{A}$  responds to the queries from  $\mathcal{A}^I$  to  $H_1$ ,  $H_2$ , Extract Private Key, Extract Partial Private Key, Request Public Key, Replace Public Key and Sign by evaluating  $H'_1$ ,  $H_2$ , Extract Private Key', Extract Partial Private Key', Request Public Key', Replace Public Key' and Sign', respectively. Let the output of  $\mathcal{A}^I$  be  $(ID_{out}, M, \sigma)$ .

(3) If  $ID_{out} = ID$ , then outputs  $(ID, M, \sigma)$ , otherwise fails.

We can see that the number of queries is the same for both  $\mathcal{A}^I$  and  $\mathcal{A}$  and the running time of  $\mathcal{A}$  is at most  $t$ , since  $\mathcal{A}^I$  may take some extra significant time analyzing the answers of queries.

Since the distributions produced by  $H'_1$ , Extract Private Key', Extract Partial Private Key', Request Public Key', Replace Public Key' and Sign' are indistinguishable from those produced by our scheme,  $\mathcal{A}^I$  learns nothing from the query results, and hence:

$$Pr[(ID_{out}, M, \sigma) \text{ is valid}] \geq \epsilon$$

Since  $H_1$  is a random oracle, the probability that the output  $(ID_{out}, M, \sigma)$  is valid, without any query to

$H'_1(ID_{out})$ , is negligible, so:

$$\begin{aligned} & Pr[ID_{out} = ID_i \text{ for some } i \mid \\ & \quad (ID_{out}, M, \sigma) \text{ is valid}] \\ & \geq 1 - \frac{1}{2^k} \end{aligned}$$

Since  $r$  is independently and randomly chosen,

$$\begin{aligned} & Pr[ID_{out} = ID_r \mid ID_{out} = ID_i \text{ for some } i] \\ & \geq \frac{1}{q_{h_1}} \end{aligned}$$

Combining these equalities, we have

$$\begin{aligned} \epsilon_1 &= Pr[ID_{out} = ID_r = ID \text{ and} \\ & \quad (ID_{out}, M, \sigma) \text{ is valid}] \\ & \geq \epsilon(1 - \frac{1}{2^k})/q_{h_1} \end{aligned}$$

and the proof is done.  $\square$

**Lemma 2.** *Let us assume that there is an adaptively chosen message and given identity Type-I attacker  $\mathcal{A}$  that makes  $q_{h_i}$  queries to the random oracle  $H_i$  ( $i = 1, 2$ ),  $q_s$  queries to the signing oracle. Assume that, within time  $t$ ,  $\mathcal{A}$  produces a forgery with probability  $\epsilon \geq 10(q_s + 1)(q_s + q_{h_2})/2^k$ . Then there exists an algorithm  $\mathcal{B}$  that is able to solve the  $q$ -SDH in  $(\mathbb{G}_1, \mathbb{G}_2)$  for  $q = q_{h_1}$  in expected time*

$$\begin{aligned} t' &\leq 120686q_{h_2} \\ & \quad \frac{t + O(q_s(\tau_{pair} + \tau_{exp})) + O(\tau_{exp}) + O(q_{h_2})}{\epsilon(1 - q_s/2^k)} \\ & \quad + O(q^2\tau_{mult}) \end{aligned}$$

where  $\tau_{mult}$  is the cost of a scalar multiplication in  $\mathbb{G}_2$ ,  $\tau_{pair}$  is the cost of a pairing evaluation, and  $\tau_{exp}$  is the cost of an exponentiation in  $\mathbb{G}_T$ .

*Proof.* This proof is similar to the proof of Lemma 2 in [3].

We will construct algorithm  $\mathcal{B}$  which takes as input  $(P, Q, \alpha Q, \alpha^2 Q, \dots, \alpha^q Q)$  and aims to find a pair  $(c, \frac{1}{c+\alpha}P)$ . In the Setup phase,  $\mathcal{B}$  builds a generator  $G \in \mathbb{G}_1$  such that it knows  $q-1$  pairs  $(w_i, \frac{1}{w_i+\alpha}G)$  for  $w_1, \dots, w_{q-1} \in_R \mathbb{Z}_p^*$ . To do so,

(1) It picks  $w_1, \dots, w_{q-1} \leftarrow_R \mathbb{Z}_p^*$  and expands  $f(z) = \prod_{i=1}^{q-1} (z + w_i)$  to obtain  $c_0, \dots, c_{q-1} \in \mathbb{Z}_p^*$  such that  $f(z) = \sum_{i=0}^{q-1} (c_i z^i)$ .

(2) It sets generators  $H = \sum_{i=0}^{q-1} c_i (\alpha^i Q) = f(\alpha)Q \in \mathbb{G}_2$  and  $G = \psi(H) = f(\alpha)P \in \mathbb{G}_1$ . It defines the public parameter  $H_{pub} = \sum_{i=1}^q c_{i-1} (\alpha^i Q) \in \mathbb{G}_2$ , so  $H_{pub} = \alpha H$ , although  $\mathcal{B}$  does not know  $\alpha$ .

(3) For  $1 \leq i \leq q-1$   $\mathcal{B}$  expands  $f_i(z) = f(z)/(z + w_i) = \sum_{i=0}^{q-2} (d_i z^i)$  and

$$\begin{aligned} & \sum_{i=0}^{q-2} d_i \psi(\alpha^i Q) \\ &= f_i(\alpha)P = \frac{f(\alpha)}{\alpha + w_i}P = \frac{1}{\alpha + w_i}G \end{aligned} \quad (5.1)$$

The pairs  $(w_i, \frac{1}{w_i+\alpha}G)$  are computed using the left member of (5.1).

$\mathcal{B}$  chooses the challenge identity  $ID^* \leftarrow_R \{0, 1\}^*$ , then returns  $ID^*$  to  $\mathcal{A}$  and the system

$$\begin{aligned} \text{params} &= \langle p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot), \psi, \\ & \quad G, H, H_{pub}, g', H_1, H_2 \rangle \end{aligned}$$

where  $H_1$  and  $H_2$  are random oracles controlled by  $\mathcal{B}$  and  $g' = e(G, H)$ .

We may assume that (1) queries to  $H_1$  are distinct, (2) any query involving an identifier  $ID$  is preceded by query to the random oracle  $H_1(ID)$  and (3) a public key for  $ID$  is already requested (Request Public Key) *before* any queries to get a secret key (Extract Private Key) and any queries to replace (Replace Public Key).

$\mathcal{B}$  initializes a counter  $l$  to 1 and generates two empty lists  $L_1$  and  $L_2$ .  $L_1$  stores tuples  $(ID, w, N, D, t)$  for the randomly chosen  $w$  (above) and public key  $N$ , partial key  $D$  and secret value  $t$  for  $ID$ . With  $L_2$ ,  $\mathcal{B}$  can control his answers to the random oracle  $H_2$  queries from  $\mathcal{A}$  (for verification of signatures).  $\mathcal{B}$  responds to the queries from  $\mathcal{A}$  in the following way:

(1)  $H_1$ -queries on an identity  $ID$ :  $\mathcal{B}$  stores  $(ID, w_l, N = \perp, D = \perp, t = \perp)$  in the list  $L_1$ , returns  $w_l$  and increments  $l$ .

(2)  $H_2$ -queries on  $(M, ID, N, r, rN)$ :  $\mathcal{B}$  retrieves  $(M, ID, N, r, rN, h_2)$  from a list  $L_2$  and returns  $h_2$ . If that tuple cannot be retrieved, it aborts.

(3) Public Key Request queries on  $ID$ :  $\mathcal{B}$  recovers  $(ID, w, N, D, t)$  from  $L_1$ . If  $N \neq \perp$  returns  $N$ . Otherwise chooses  $t \leftarrow_R \mathbb{Z}_p^*$ , computes  $N = g^t$ , updates the tuple in  $L_1$  with  $N$  and  $t$ ; returns  $N$ .

(4) Partial Key Extraction queries on  $ID$ :  $\mathcal{B}$  recovers  $(ID, w, N, D, t)$  from  $L_1$ , update and returns  $D = (1/(\alpha + w))G$ , computed previously.

(5) Secret Key Extraction queries on  $ID$ :  $\mathcal{B}$  retrieves  $(ID, w, N, D, t)$  from  $L_1$ . If  $D = \perp$  it makes first a Partial Key Extraction query. Returns  $(D, t)$ .

(6) Public Key Replacement queries on  $ID$  with  $N'$ :  $\mathcal{B}$  recovers  $(ID, w, N, D, t)$  from  $L_1$ . Updates the tuple with  $N = N'$  and  $t = \perp$  in  $L_1$ .

(7) Signature queries on  $ID, M$ :  $\mathcal{B}$  chooses  $S \leftarrow_R \mathbb{G}_1$  and  $h \leftarrow_R \mathbb{Z}_p^*$ . Computes  $r = e(S, Q_{ID})e(G, H)^{t-h}$ , where  $Q_{ID} = H_1(ID)H + H_{pub}$  and  $t$  could be extracted from  $L_1$  (if the public key was not replaced) or  $\mathcal{A}$  submits a  $t'$  corresponding to the replaced public key  $N$  (but if  $ID = ID^*$  or  $N \neq g^{t'}$ , it aborts).  $\mathcal{B}$  defines  $H_2(M, ID, N, r, rN)$  as  $h$ , including  $(M, ID, N, r, rN, h)$  in the list  $L_2$  (it aborts if this tuple is already defined for  $h' \neq h$ ). Returns  $\sigma = (h, S)$ .

We have explained how  $\mathcal{B}$  simulates an environment for an attack of  $\mathcal{A}$ .

If  $\mathcal{B}$  does not abort during the simulation, then the  $\mathcal{A}$ 's view is the same as its view in a real attack. This is true since the values  $w$  and  $h$  are independently and randomly chosen (at preparation phase and signature queries processing), and  $H_1()$  and  $H_2()$  are random oracles. Moreover,  $\mathcal{B}$  can abort (without an earlier abortion of  $\mathcal{A}$ ) only when, in a signature query, the same value  $h$  is chosen twice. The probability of this event is at most  $q_s/2^k$ , since  $h$  is randomly chosen. In other words,  $\Pr[\mathcal{B} \text{ does not fail}] \geq \epsilon(1 - q_s/2^k)$ .

Our purpose now is to apply the Forking Lemma (Theorem 3 in [12]). This Lemma essentially says: consider a scheme producing signatures of the form  $(M, r, h, S)$ , where each of  $r, h, S$  corresponds to one of the three moves of an honest-verifier zero-knowledge protocol. Let us assume that a chosen message attacker

$\mathcal{F}$  forges a signature  $(M, r, h, S)$  in expected time  $t$  with probability  $\epsilon \geq 10(q_s + 1)(q_s + q_h)/2^k$  when making  $q_s$  signature queries and  $q_h$  random oracle calls. If the triples  $(r, h, S)$  can be simulated without knowing the private key, then there exists a Turing machine  $\mathcal{F}'$  that uses  $\mathcal{F}$  to produce two valid signatures  $(M, r, h_1, S_1)$  and  $(M, r, h_2, S_2)$ , with  $h_1 \neq h_2$ , in expected time  $t' \leq 120686q_h t/\epsilon$  [3].

In our setting, from  $\mathcal{A}$  we build an algorithm  $\mathcal{A}'$  that replays  $\mathcal{A}$  using  $ID^*$  and

$$\text{params} = \langle p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(), \psi, G, H, H_{pub}, g', H_1, H_2 \rangle$$

until two forgeries  $(M^*, r, h_1, S_1)$  and  $(M^*, r, h_2, S_2)$ , with  $h_1 \neq h_2$ , could be obtained.

Then  $\mathcal{B}$  runs  $\mathcal{A}'$  to obtain two valid signatures  $(M^*, r, h_1, S_1)$  and  $(M^*, r, h_2, S_2)$  for  $ID^*$  and for the same message  $M^*$  and commitment  $r$ .  $\mathcal{B}$  recovers  $(ID^*, w^*)$  from the list  $L_1$ . Since the two signatures are valid for the same  $r$  and the public key for  $ID^*$  was not replaced ( $t^*$  has never changed), we have:

$$e(S_1, Q_{ID^*})e(G, H)^{t^*-h_1} = e(S_2, Q_{ID^*})e(G, H)^{t^*-h_2}$$

where  $Q_{ID^*} = H_1(ID^*)H + H_{pub} = (w^* + \alpha)H$ . Then,

$$e((h_1 - h_2)^{-1}(S_1 - S_2), Q_{ID^*}) = e(G, H),$$

and  $T^* = (h_1 - h_2)^{-1}(S_1 - S_2) = (1/(\alpha + w^*))G$ . From  $T^*$ ,  $\mathcal{B}$  can proceed as in [3][6] to extract  $\sigma^* = (1/(\alpha + w^*))P$ :

$\mathcal{B}$  first obtains  $\gamma_{-1}, \gamma_0, \dots, \gamma_{q-2} \in \mathbb{Z}_p^*$ , for which  $f(z)/(z + w^*) = \gamma_{-1}/(z + w^*) + \sum_{i=0}^{q-2} (\gamma_i z^i)$  and computes

$$\sigma^* = \frac{1}{\gamma_{-1}} \left[ T^* - \sum_{i=0}^{q-2} \gamma_i \psi(\alpha^i Q) \right] = \frac{1}{\alpha + w^*} P$$

Then  $\mathcal{B}$  returns  $(w^*, \sigma^*)$  as the result for that given instance for  $q$ -SDH.

To compute the complexity of  $\mathcal{B}$ 's running time, consider:

(1)  $\mathcal{A}$  forges a signature in a time  $t$  with probability  $\epsilon \geq 10(q_s + 1)(q_s + q_h)/2^k$  (to use the Forking Lemma);

(2)  $\Pr[\mathcal{B} \text{ does not fail}] \geq \epsilon(1 - q_s/2^k)$  (as shown before).

(3) the cost of the preparation phase is  $O(q_{h_1}^2 \tau_{mult})$ , where  $\tau_{mult}$  is the cost of a scalar multiplication in  $\mathbb{G}_2$ ;

(4) the cost of processing  $H_2$  queries is  $O(q_{h_2})$  since only the list  $L_2$  is involved;

(5) the number of queries for partial and secret extractions, replacements and requests of public keys is  $O(q_{h_1})$ . Since those queries, as well as  $H_2$ -queries, involve only operations on the list  $L_1$  we conclude that the cost of queries for partial and secret key extractions, replacements of public keys and  $H_2$ -queries is  $O(\log(q_{h_1}))$ ;

(6) the number of queries for public key requests is  $O(q_{h_1})$ . Since those queries involve exponentiation in  $\mathbb{G}_T$  its complexity is  $O(q_{h_1} \tau_{exp})$  where  $\tau_{exp}$  denotes time for one exponentiation in  $\mathbb{G}_T$ ;

- (7) the cost of processing signature queries is  $O(q_s(\tau_{pair} + \tau_{exp}))$ , where  $\tau_{pair}$  is the cost of a pairing evaluation;  
 (8)  $q = q_{h_1}$ .

Hence  $\mathcal{B}$  solves  $q$ -SDH in  $(\mathbb{G}_1, \mathbb{G}_2)$  in expected time:

$$t' \leq \frac{120686q_{h_2} t + O(q_s(\tau_{pair} + \tau_{exp})) + O(q\tau_{exp}) + O(q_{h_2})}{\epsilon(1 - q_s/2^k)} + O(q^2\tau_{mult})$$

The proof is now done.  $\square$

**Lemma 3.** *If there is an algorithm  $\mathcal{A}^{II}$  for an adaptively chosen message and identity Type-II attack against our CL-PKS scheme with running time  $t$  and advantage  $\epsilon$ , then there is an algorithm  $\mathcal{B}$  that is able to solve the Bilinear Pairing Inversion Problem (BPI) in  $\mathbb{G}_T$  in expected time  $t' \leq t$ , with the same advantage  $\epsilon$ .*

*Proof.* This proof is similar to the previous two Lemmas. First, as in the proof of Lemma 1, we construct an attacker  $\mathcal{A}$ , challenged on a *given identity*, if our CL-PKS scheme is under a Type-II attacker. Second, as in the proof of Lemma 2, we prove that if there is such a polynomial time adversary  $\mathcal{A}$  then there is a polynomial time (on  $k$ ) algorithm to solve the BPI Problem in  $\mathbb{G}_T$ , which is a hard problem (see Section 2.2).  $\square$

## 6. Conclusions

We have constructed a CL-PKS scheme that does not allow key escrow by the KGC, based on asymmetric bilinear pairing functions. We analyzed both its efficiency and security. It is more efficient than previously published CL-PKS schemes, with shorter signatures, and we proved it is strong against adaptively chosen message attacks.

## References

- [1] AL-RIYAMI, S. S.; PATERSON, K. G. Certificateless Public Key Cryptography. 2003. Cryptology ePrint Archive, Report 2003/126. <http://eprint.iacr.org/>.
- [2] AL-RIYAMI, S. S.; PATERSON, K. G. C: CBE from CL-PKS: A generic construction and efficient schemes. In: Public Key Cryptography - PKC 2005. [S.l.: s.n.], 2005. pp. 398-415.
- [3] BARRETO, P.S.L.M., B. LIBERT, N. McCULLAGH, J. QUISQUATER. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps, LNCS vol. 3788, Nov. 2005, pp 515-532.
- [4] BARRETO, P.S.L.M., NAEHRIG, M. Pairing-friendly elliptic curves of prime order. In: Proceedings of SAC 2005, Lec. Notes in Comp. Sci., Springer, vol. 3897, pp 319-331.
- [5] BONEH, D.; FRANKLIN, M. K. Identity-based encryption from the Weil pairing. In: CRYPTO 01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. London, UK: Springer-Verlag, 2001. pp. 213-229. ISBN 3-540-42456-3. <http://eprint.iacr.org/2001/090/>.
- [6] BONEH, D.; BOYEN, X. Short signatures without random oracles. In Proceedings of the Eurocrypt 2004, Springer LNCS 3027, pp. 53-76, 2004.
- [7] BONEH, D.; BOYEN, X. Efficient selective-ID secure identity based encryption without random oracles, In Eurocrypt 2004, volume 3027 of LNCS, pages 223-238, Springer, 2004.
- [8] CHA, J. C., CHEON, J. H. An identity-based signature from gap Diffie-Hellman groups, In PKC'03, volume 2567 of LNCS, pp18-30, Springer, 2003.
- [9] CHEON, J. H.; LEE, D. H. Diffie-Hellman Problems and Bilinear Maps. 2002. Cryptology ePrint Archive, Report 2002/117. <http://eprint.iacr.org/>.
- [10] HUANG, X.; SUSILO, W.; MU, Y.; ZHANG, F. On the security of certificateless signature schemes from Asiacrypt 2003, Proceedings of the Cryptology and Network Security - CANS - 2005, LNCS 3810, pp 13-25, 2005, Springer.
- [11] MIYAJI, A.; NAKABAYASHI, M.; TAKANO, S. New explicit conditions of elliptic curve traces for FR-reduction. IEICE Transactions on Fundamentals, vol E84-A(5):1234-1243, 2001.
- [12] POINTCHEVAL, D., STERN, J. Security arguments for digital signatures and blind signatures, Journal of Cryptology, 13(3):361-396, 2000.
- [13] SHAMIR, A. Identity-based cryptosystems and signature schemes. In: Proceedings of CRYPTO 84 on Advances in cryptology. New York, NY, USA: Springer-Verlag New York, Inc., 1984. pp. 47-53. ISBN 0-387-15658-5.
- [14] SMART, N. P.; VERKAUTEREN, F. On computable isomorphisms in efficient pairing based systems. Cryptology ePrint Archive Report 2005/116, 2005. <http://eprint.iacr.org/2005/116>.
- [15] SCOTT, M.; BARRETO, P. S. L. M. Compressed pairings. In: . Springer, 2004. (Lecture Notes in Computer Science, vol. 3152), pp. 140-156. Available as <<http://eprint.iacr.org/2004/032>>
- [16] YACOBI, Y., A Note on the Bi-linear Diffie-Hellman assumption. Cryptology ePrint Archive Report 2002/113, 2002, <http://eprint.iacr.org/2002/113>
- [17] ZHANG, Z.; WONG, D. S.; XU, J.; FENG, D. Certificateless Public Key Signature: Security Model and Efficient Construction. In: Proceedings of the 4th. Internat'l Conference on Applied Cryptography and Network Security, June 6-9, 2006, Singapore. Lec. Notes in Comp. Sci. vol. number 3989, by Springer.

## Appendix: The Security Model

This Appendix defines the types of adversary and games (as defined in [1]) used in Section 5.

**Game-I:** This is the game in which an adversary of Type I  $\mathcal{A}^I$  interacts with the *Challenger*:

**Phase 1-I** The *Challenger* runs  $\text{Setup}(1^k)$  to generate **master-key** and **params**. *Challenger* gives **params** to  $\mathcal{A}^I$  and keeps safely the **master-key**.

**Phase 2-I**  $\mathcal{A}^I$  makes queries of the following types:

(1) **Extract Partial Private Key** For a given  $ID_A$ , *Challenger* computes  $D_A = \text{Partial-Private-Key-Extract}(\text{params}, \text{master-key}, ID_A)$  and returns it to  $\mathcal{A}^I$ .

(2) **Extract Private Key** For a given  $ID_A$ , *Challenger* computes first  $D_A = \text{Partial-Private-Key-Extract}(\text{params}, \text{master-key}, ID_A)$  and  $t_A = \text{Set-Secret-Value}(\text{params}, ID_A)$ . Returns  $S_A = \text{Set-Private-Key}(\text{params}, ID_A, D_A, t_A)$  to  $\mathcal{A}^I$ .

(3) **Request Public Key** For a given  $ID_A$ , *Challenger* computes first  $D_A = \text{Partial-Private-Key-Extract}(\text{params}, \text{master-key}, ID_A)$  and  $t_A = \text{Set-Secret-Value}(\text{params}, ID_A)$ . Returns  $N_A = \text{Set-Public-Key}(\text{params}, ID_A, t_A)$  to  $\mathcal{A}^I$ .

(4) **Replace Public Key** For a given  $ID_A$ ,  $\mathcal{A}^I$  may replace a public key  $N_A$  with a value chosen by him.  $\mathcal{A}^I$  is not required to provide the corresponding secret value.

(5) **Signing Queries** For a given  $ID_A$ , and chosen message  $M$ , if  $N_A$  has not been replaced, *Challenger* gets  $S_A$  from its “query-answer” list and returns  $\sigma = \text{CL-Sign}(\text{params}, M, ID_A, D_A, t_A)$ . If  $N_A$  has been replaced,  $\mathcal{A}^I$  may submit the secret value  $t_A$  corresponding to the replaced public key to the signing oracle.

**Phase 3-I**  $\mathcal{A}^I$  outputs a message  $M^*$  and a signature  $\sigma^*$  for the identity  $ID^*$  with  $N_A^*$ .  $ID^*$  cannot be an identity for which the private key has been extracted, and  $ID^*$  cannot be an identity for which both the public key has been replaced and the partial private key has been extracted. Moreover,  $M^*$  should not have been queried to the signing oracle with  $ID^*$  and  $N_A^*$ . However, in the case  $N_A^*$  is different from the original public key of the entity with  $ID^*$ ,  $\mathcal{A}^I$  should not have submitted the corresponding secret value if it has not made signing queries for  $ID^*$  and  $N_A^*$ .

**Game-II** This is the game in which an adversary of Type II  $\mathcal{A}^{II}$  interacts with the *Challenger*:

**Phase 1-II** The *Challenger* runs  $\text{Setup}(1^k)$  to generate **master-key** and **params**, both returned to  $\mathcal{A}^{II}$ .

**Phase 2-II**  $\mathcal{A}^{II}$  makes queries of the following types

(1) **Extract Partial Private Key** For a given  $ID_A$ ,  $\mathcal{A}^{II}$  computes  $D_A = \text{Partial-Private-Key-Extract}(\text{params}, \text{master-key})$ .

(2) **Extract Private Key** For a given  $ID_A$ , *Challenger* computes first  $D_A = \text{Partial-Private-Key-Extract}(\text{params}, \text{master-key}, ID_A)$  and  $t_A = \text{Set-Secret-Value}(\text{params}, ID_A)$ . Returns  $S_A = \text{Set-Private-Key}(\text{params}, ID_A, D_A, t_A)$  to  $\mathcal{A}^{II}$ .

(3) **Request Private Key** For a given  $ID_A$ , *Challenger* computes first  $D_A = \text{Partial-Private-Key-Ex-}$

$\text{tract}(\text{params}, \text{master-key}, ID_A)$  and  $t_A = \text{Set-Secret-Value}(\text{params}, ID_A)$ . Returns  $N_A = \text{Set-Public-Key}(\text{params}, ID_A, t_A)$  to  $\mathcal{A}^{II}$ .

(4) **Signing Queries** For a given  $ID_A$ , a chosen message  $M$ , *Challenger* gets  $S_A$  from its “query-answer” list and returns  $\sigma = \text{CL-Sign}(\text{params}, M, ID_A, S_A)$ .

**Phase 3-II**  $\mathcal{A}^{II}$  outputs a message  $M^*$  and a signature  $\sigma^*$  for the identity  $ID^*$  with  $N_A^*$ .  $ID^*$  cannot be an identity for which the private key has been extracted. Moreover,  $M^*$  should not have been queried to the signing oracle with  $ID^*$  and  $N_A^*$ .

Informally we say that an adversary  $\mathcal{A}$  ( $\mathcal{A}^I$  or  $\mathcal{A}^{II}$ ) achieves success in the above games if

$$\text{CL-Verify}(\text{params}, ID^*, N_A^*, M^*, \sigma^*) = 1$$

We say that a CL-PKS scheme is *existentially unforgeable against adaptive chosen message attacks* if any probabilistic polynomial time on  $k$  adversary  $\mathcal{A}$  has a negligible probability in gaining success. Formally:

**Definition 4.** (EUF-CMA of a CL-PKS, [17]) A CL-PKS scheme  $\mathcal{S}$  is Existentially Unforgeable against Type I or II Adaptive Chosen Message Attacks if and only if the probability of success of any polynomial on  $k$  time bounded Type I (resp. II) adversary  $\mathcal{A}$  in the Game I (resp. II) as described above is negligible, i.e., given a security parameter  $k$ :

$$\forall \epsilon > 0 : \Pr_{\mathcal{A}, \mathcal{S}, k} \{\text{success}\} \leq \epsilon$$