Security Model for Certificateless Aggregate Signature Schemes

Lei Zhang  
College of Mathematics and Computer Science  
Nanjing Normal University  
Nanjing, P.R. China  
lei_zhangzl@126.com

Futai Zhang  
College of Mathematics and Computer Science  
Nanjing Normal University  
Nanjing, P.R. China  
zhangfutai@njnu.edu.cn

Abstract

Gong, Long, Hong and Chen defined the security model of certificateless aggregate signature schemes for the first time. However, there are some weaknesses exist in their model. In this paper, we point out some drawbacks of the security model of Gong et al.’s and present a new one. A certificateless aggregate signature scheme that is provably secure in our model is also presented. The security of our scheme is proved based on the intractability of the computational Diffie-Hellman problem in the random oracle model.

Index Terms—certificateless cryptography; aggregate signature; certificateless aggregate signature; random oracle model.

1 Introduction

The concept of aggregate signature was introduced by Boneh, Gentry, Lynn and Shacham [3] in Eurocrypt 2003. Aggregate signatures [3, 4] have many applications. For example, they are used for reducing the size of certificate verification chains (by aggregating all signatures in the chain) and for reducing message size in secure routing protocols such as Secure Border Gateway Protocol (SBGP) [9]. They are also useful in some other special areas where the signatures on many different messages generated by many different users need to be compressed.

In traditional public key cryptosystems, the biggest problem is the requirement of a large amount of computing and storage cost to manage certificates. Thanks to the introduction of the Identity-Based public key cryptography (ID-PKC) [11], this condition has greatly changed. In ID-PKC, the public key of a user is just his identity such as his telephone number or email address. It eliminates the need of public key certificates to authenticate users’ public keys. However, the key escrow problem is inherent in this setting. To overcome the key escrow problem of ID-PKC, Al-Riyami and Paterson [1] invented a new paradigm called certificateless public key cryptography (CL-PKC). In CL-PKC, a third party called Key Generation Center (KGC) which is used to help a user to generate his secret key is used. However, the KGC does not access to the user’s whole private key, it only provides the partial private key for the user. The full private key is finally generated by the user who makes use of the partial private key obtained from the KGC and the secret information chosen by himself.

Related Works and Motivation. The first certificateless signature (CLS) scheme was presented by Al-Riyami and Paterson [1], however, no formal cryptanalysis was given to this scheme. In 2005, Huang et al. [7] pointed out a secure drawback of the CLS scheme in [1]. They also defined the security model of CLS schemes in the same paper. But the model in [7] did not fully catch the ability of the type I adversaries. A typical example is Yap et al.’s scheme [12]. This scheme is provably secure in the model of Huang et al.’s. Unfortunately, it was broken by Park et al. [10] and Zhang et al. [14] respectively. So a CLS scheme which is provably secure in this model may actually be insecure. Later, Zhang et al. [15] improved the security model of CLS schemes and presented a more efficient CLS scheme. In [6], Hu et al. further developed the security model of CLS schemes. In their model they give Type I/II adversaries very strong abilities that is the Type I/II adversaries can obtain some message-signature pairs which are valid under the public key chosen by themselves without supplying the secret value corresponding to the public key. This type of adversaries is called super Type I/II adversaries in [8]. Recently, security models in certificateless public key cryptography which treat Type I/II adversaries as super adversaries are the most popular ones.

Certificateless public key cryptography inherits the excellent property of ID-PKC (does not require any certificate), while without suffering the key escrow problem which troubles ID-PKC. So, it is interesting to study secure and efficient constructions of aggregate signatures in CL-PKC. With the technique of certificateless aggregated signature (CLAS) in hand, one can in the former cases, ag-
aggregate many different certificateless signatures into a single CLAS, and hence effectively reduce the message size and verification cost. Further more, the escrow free property of CL-PKC makes it impossible for the malicious KGC to forge any valid CLAS without being detected. In [5], Gong, Long, Hong and Chen defined the security model of CLAS schemes for the first time. However, their model is closely related to the security model in [7]. Hence, their model does not essentially capture the most powerful attack of type I/II adversaries. In the same paper, they also presented two CLAS schemes. But their securities are proved in their loose model.

Our Contribution. In this paper, we point out some drawbacks of the security model in [5] and present a new one. In our security model of CLAS schemes, we treat Type I/II adversaries as super adversaries. A CLAS scheme which can be proved in our security model is presented as well.

2 Preliminaries

2.1 Bilinear Maps

Let $G_1$ be an additive group of prime order $q$ and $G_2$ be a multiplicative group of the same order. $e: G_1 \times G_1 \rightarrow G_2$ is called a bilinear map if it satisfies the following properties:

1. Bilinear: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$, $a, b \in \mathbb{Z}_q^*$.
2. Non-degeneracy: There exists $P, Q \in G_1$ such that $e(P, Q) \neq 1$.
3. Computable: There exists an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G_1$.

Computational Diffie-Hellman (CDH) Problem: Given a generator $P$ of the cyclic additive group $G$ whose order is $q$, and given $(aP, bP)$ for unknown $a, b \in \mathbb{Z}_q^*$, to compute $abP$.

3 Security Model

3.1 Description of the Model

Two types of adversaries are considered in CL-PKC — Type I adversaries and Type II adversaries. A Type I adversary $A_I$ does not have access to the master-key, but he has the ability to replace the public key of any user with a value of his choice. While a Type II Adversary $A_{II}$ has access to the master-key but cannot replace the target user’s public key. More details can be found in [6, 8, 13].

To define the security of a CLAS scheme (For the formal definition of a CLAS scheme please refer to [5]), we first define the following oracles.

- Partial-Private-Key-Oracle: The input of this oracle is an identity $ID_i$. The output of this oracle is the partial private key $D_i$ for $ID_i$.
- Public-Key-Oracle: The input of this oracle is an identity $ID_i$. The output of this oracle is the public key for $ID_i$.
- Secret-Value-Oracle: The input of this oracle is an identity $ID_i$. The output of this oracle is the secret value $x_i$ for $ID_i$ (It outputs $\bot$, if the user’s public key has been replaced).
- Public-Key-Replacement-Oracle: The input of this oracle is an identity $ID_i$ and a public key $P'_i$. This oracle is used to replace the public key of $ID_i$ to the value $P'_i$.
- Sign-Oracle: The input of this oracle is an identity $ID_i$, a public key $P_i$, a message $M_i$. The output of this oracle is a valid signature $\sigma$ on message $M_i$ under identity $ID_i$ and public key $P_i$. (Note that the secret value which is used to generate $P_i$ is not required for this oracle).

The security of a CLAS scheme is modeled via the following EUF-CLAS-CMA2 game between a challenger $C$ and an adversary $A \in \{A_I, A_{II}\}$. And we treat $A_I$ and $A_{II}$ as super adversaries. Now we begin to describe the EUF-CLAS-CMA2 game. This game is separated into three stages — Setup, Attack and Forgery. The description of each stage comes as follows.

Setup: $C$ runs the Setup algorithm of a CLAS scheme, takes as input a security parameter $\ell$ to obtain a master-key and the system parameter list params. If $A$ is a type I adversary, then $C$ sends params to $A$ while keeps the master-key secretly. Otherwise $A$ is a type II adversary; $C$ sends params and master-key to $A$.

Attack: In this stage, the adversary $A$ (no matter type I or II) has the ability to access the following oracles which are controlled by $C$: Partial-Private-Key-Oracle, Public-Key-Oracle, Secret-Value-Oracle, Public-Key-Replacement-Oracle, Sign-Oracle.

Forgery: Finally, $A$ outputs a set of $n$ messages $L^*_M = \{M^*_1, ..., M^*_n\}$, $n$ users’ identities $L^*_D = \{ID^*_1, ..., ID^*_n\}$ and their corresponding public keys $L^*_P = \{P^*_1, ..., P^*_n\}$ and an aggregate signature $\sigma^*$.

When $A$ is a type I adversary $A_I$, we say that $A$ wins above game, if the following requirements are satisfied.

1. $\sigma^*$ is a valid aggregate signature on messages $\{M^*_1, ..., M^*_n\}$ under identities $\{ID^*_1, ..., ID^*_n\}$ and corresponding public keys $\{P^*_1, ..., P^*_n\}$ chosen by $A$. 

2. One of the identities, without loss of generality, say \(ID^*_i \in L^*_D\) has not been submitted to the Partial-Private-Key-Oracle queries. And \((ID^*_i, P^*_i, M^*_i)\) has never been submitted to the Sign-Oracle.

When \(A\) is a type II adversary \(A_{II}\), we say that \(A\) wins above game, if the following requirements are satisfied.

1. \(\sigma^*\) is a valid aggregate signature on messages \(\{M^*_1, ..., M^*_n\}\) under identities \(\{ID^*_1, ..., ID^*_n\}\) and corresponding public keys \(\{P^*_1, ..., P^*_n\}\) chosen by \(A\).

2. One of the identities, without loss of generality, say \(ID^*_i \in L^*_D\) has not been submitted to the Public-Key-Replacement-Oracle and Secret-Value-Oracle. And \((ID^*_1, P^*_1, M^*_1)\) has never been submitted to the Sign-Oracle.

Definition 1: A CLAS scheme is existentially unforgeable under adaptive chosen-message attack iff the success probability of any polynomially bounded adversary \(A\) winning the above game is negligible.

3.2 Differences from the Old Definition

Here we clearly illustrate the main improvement of our security model (compared with the security model in [5]).

Firstly, we allow the type II adversary to have the ability to access the Public-Key-Replacement-Oracle. This comes from the fact that a type II adversary can cooperate with some users to help him to forge a significative CLAS.

Secondly, in [5], the adversary \(A\) is restricted not to submit \((ID^*_1, +, *)\) to the Sign-Oracle. In other words, \(A\) is not allowed to obtain any signature form the user whose identity is \(ID^*_1\). (Essentially, this attack equals to “no message attack”). This is an unreasonable restriction. The Sign-Oracle which is controlled by \(C\) should answer all of \(A\)’s signature queries expect for the query on \((ID^*_1, P^*_1, M^*_1)\). And in their model, it implies that when \(A\) replace the public key of \(ID^*_1\), he should also submit the corresponding secret value to \(C\) (Actually, they use the technique in [12] to prove the security of their schemes. However, this technique is improper [10, 14]). This is another unreasonable restriction. This is because \(A\) can choose a public key which he does not know the corresponding secret value as a user’s new public key [15].

Finally, in the Forgery stage, the outputs of ours are different from that of in [5]. In our model, the adversary \(A\) should only output a valid aggregate signature \(\sigma^*\) on messages \(\{M^*_1, ..., M^*_n\}\) under identities \(\{ID^*_1, ..., ID^*_n\}\) and corresponding public keys \(\{P^*_1, ..., P^*_n\}\). While in [5], they require \(A\) to output a valid signature \(\sigma^*\) on \(M^*_i\) under identity \(ID^*_i\) and corresponding public key \(P^*_i\) (the clearer evidence can be found in their Theorem 1). In other words, they mean that if a signature scheme is secure, then an aggregate signature scheme based on this signature scheme is secure. However this deduction is incorrect. A typical example can be found in [4].

4 A New Construction

In this section, we present a CLAS scheme. This CLAS scheme is based on the CLS scheme in [15].

- Setup: On input a security parameter \(\ell\), the KGC specifies \(G_1, G_2, e\), as described in Section II. The KGC also chooses a random \(\lambda \in \mathbb{Z}_q^\star\) as the master-key and sets \(P_T = \lambda P\), chooses cryptographic hash functions \(H_1, H_2, H_3 : \{0, 1\}^* \rightarrow G_1\). The system parameter list is \(\text{params} = (G_1, G_2, e, P, P_T, H_1, H_2, H_3)\).

- Partial-Private-Key-Extract: This algorithm accepts \(\lambda\) and a user’s identity \(ID_i\). It outputs the partial private key \(D_i = \lambda Q_i\), where \(Q_i = H_1(ID_i)\).

- Set-Secret-Value: This algorithm takes as input \(\text{params}\) and a user’s identity \(ID_i\). It selects a random \(x_i \in \mathbb{Z}_q^\star\) and outputs \(x_i\) as the user’s secret value.

- Set-Private-Key: This algorithm accepts \(\text{params}\), a user’s identity \(ID_i\), partial private key \(D_i\) and secret value \(x_i \in \mathbb{Z}_q^\star\). The output of the algorithm is the private key \(S_i = (x_i, D_i)\) of the user.

- Set-Public-Key: This algorithm accepts \(\text{params}\), a user’s identity \(ID_i\) and this user’s secret value \(x_i \in \mathbb{Z}_q^\star\). It produces the user’s public key \(P_i = x_i P\).

- Sign: To sign a message \(M_i\) using the private key \(S_i = (x_i, D_i)\), the signer chooses a random \(r_i \in \mathbb{Z}_q^\star\), computes \(R_i = r_i P, F_i = H_2(M_i, ID_i, P_i, R_i), U_i = H_3(M_i, ID_i, F_i), V_i = D_i + r_i F_i + x_i U_i\), outputs \(\sigma_i = (R_i, V_i)\) as the signature on \(M_i\).

- Verify: To verify a signature \(\sigma_i\) on a message \(M_i\) for an identity \(ID_i\) and public key \(P_i\), the verifier computes \(F_i = H_2(M_i, ID_i, P_i, R_i), U_i = H_3(M_i, ID_i, P_i), Q_i = H_1(ID_i)\). It checks \(e(V_i, P_T) = e(Q_i, P_T) e(F_i, R_i) e(U_i, P_i)\). If it holds, the verifier outputs true; otherwise outputs \(\perp\).

- Aggregate: On input an aggregating set of \(n\) users’ identities \(\{ID^*_1, ..., ID^*_n\}\) and corresponding public keys \(\{P^*_1, ..., P^*_n\}\), and message-signature pairs \((M_1, \sigma_1 = (R_1, V_1)), ..., (M_n, \sigma_n = (R_n, V_n))\) from \(ID_1, ..., ID_n\) respectively, compute \(V = \sum_{i=1}^{n} V_i\). The aggregate signature is \(\sigma = (R_1, ..., R_n, V)\).
• Aggregate Verify: To verify an aggregate signature \( \sigma = (R_1, \ldots, R_n, V) \) signed by \( n \) users on messages \( \{M_1, \ldots, M_n\} \) under identities \( \{ID_1, \ldots, ID_n\} \) and corresponding public keys \( \{P_1, \ldots, P_n\} \), the verifier performs the following steps.

1. For all \( i \in [1, n] \), compute \( Q_i = H_1(ID_i), F_i = H_2(M_i, ID_i, P_i, R_i), U_i = H_3(M_i, ID_i, P_i) \).
2. Check \( e(V, P) \overset{?}{=} e(\sum_{i=1}^{n} Q_i, P_T) \prod_{i=1}^{n} e(F_i, R_i)e(U_i, P_i) \). If the equation holds, output true; otherwise, output \( \bot \).

4.1 Comparison

We compare our scheme with the schemes in [5]. For simplicity, we only list the most costly operation Pairing Operation (\( P \)). We use the notation SL meaning signature length, PKL meaning public key length, \( |G_1| \) meaning the length of a point in \( G_1 \).

Table 1. Comparison of Three CLS Schemes

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Type</th>
<th>Verify</th>
<th>Size</th>
<th>PKL</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Scheme in [5]</td>
<td>Off</td>
<td>1,195</td>
<td>812</td>
<td>237</td>
<td>Unsecure</td>
</tr>
<tr>
<td>Second Scheme in [5]</td>
<td>Off</td>
<td>1,185</td>
<td>822</td>
<td>237</td>
<td>Unsecure</td>
</tr>
<tr>
<td>Our Scheme</td>
<td>Off</td>
<td>1,275</td>
<td>822</td>
<td>237</td>
<td>Unsecure</td>
</tr>
</tbody>
</table>

It is easy to see that our CLAS scheme is much faster than the schemes in [5] (Mostly, that is because our CLAS do not need to verify the public keys of the users in an aggregate set). And in CL-PKC, the signer usually sends his public key with his signature together to the verifier. So, among the three CLAS schemes, the total length of the output of our CLAS scheme (SL+PKL) is the shortest one.

4.2 Security Proof

Theorem 1: Our CLAS scheme is unforgeable against a super type I adversary in the random oracle model [2] assuming the CDH problem in \( G_1 \) is intractable.

Proof. Let \( C \) be a CDH attacker who receives a random instance \( (P, aP, bP) \) and has to compute the value of \( abP \) in \( G_1 \). \( A \) is a type I adversary who interacts with \( C \) as defined in Section 3 and he can break our scheme with probability \( \epsilon \) in time \( \tau \). We show that how \( C \) can use \( A \) to solve the CDH problem, i.e. compute \( abP \) with probability \( \epsilon \) in time \( \tau' \). We show that how \( C \) can use \( A \) to solve the CDH problem, i.e. compute \( abP \) with probability \( \epsilon \) in time \( \tau' \).

Setup stage: \( C \) sets \( P_T = aP \) and chooses \( \text{params} = (G_1, G_2, e, P, P_T, H_1, H_2, H_3) \), sends \( \text{params} \) to \( A \).

Attack stage: As defined in Section 3, \( A \) can ask \( C \) following oracles’ queries and we consider hash functions \( H_1, H_2 \) and \( H_3 \) as random oracles. We assume that \( A \) can access the Partial-Private-Key-Oracle at most \( q_P \) times. And we assume that \( A \) can access all kinds of oracles at most \( q_T \) times.

\( H_1 \)-Oracle: \( C \) maintains a list \( H_1^{\text{list}} \) of tuples \( (ID, \mu, Q_{ID}, D_{ID}, c) \) to avoid collision. For input \( ID_i \), \( C \) first picks \( \mu_i \in Z_q^* \) at random then flips a coin \( c_i \in \{0,1\} \) that yields 1 with probability \( \delta \) and 0 with probability \( 1 - \delta \). If \( c_i = 1 \), \( C \) sets \( Q_i = \mu_i bP, D_i = \bot \), returns \( Q_i \) as answer and adds \( (ID_i, \mu_i, Q_i, D_i, c_i) \) to \( H_1^{\text{list}} \). Else, \( c_i = 0 \), \( C \) sets \( Q_i = \mu_i P, D_i = \mu_i P_T \), adds \( (ID_i, \mu_i, Q_i, D_i, c_i) \) to \( H_1^{\text{list}} \) and returns \( Q_i \) as answer.

\( H_2 \)-Oracle: \( C \) keeps a list \( H_2^{\text{list}} \) of tuples \( (M, ID, P_{ID}, R, \nu, F) \). Whenever \( A \) submits \( (M_i, ID_i, P_i, R_i) \) to \( H_2 \)-Oracle, \( C \) randomly choose \( \nu_i \in Z_q^* \), sets \( F_i = \nu_i P \), add \( (M_i, ID_i, P_i, R_i, \nu_i, F_i) \) to \( H_2^{\text{list}} \) and return \( F_i \) as answer.

\( H_3 \)-Oracle: \( C \) keeps a list \( H_3^{\text{list}} \) of tuples \( (M, ID, P_{ID}, \pi, U) \). Whenever \( A \) issues a query \( (M_i, ID_i, P_i) \) to \( H_3 \)-Oracle, \( C \) picks \( \pi_i \in Z_q^* \) at random, sets \( U_i = \pi_i P \), returns \( U_i \) as answer and adds \( (M_i, ID_i, P_i, \pi_i, U_i) \) to \( H_3^{\text{list}} \).

Partial-Private-Key-Oracle: On input \( ID_i, C \) first submits \( ID_i \) to \( H_1 \)-Oracle, then searches \( H_1^{\text{list}} \) for a tuple \( (ID_i, \mu_i, Q_i, D_i, c_i) \). If \( c_i = 1 \), \( C \) aborts; else, returns \( D_i \).

Public-Key-Oracle: \( C \) maintains a list \( K^{\text{list}} \) of tuples \( (ID, x, P_T) \). For input \( ID_i \), \( C \) selects \( x_i \in Z_q^* \) randomly, sets \( F_i = x_i P \), returns \( P_i \) and adds \( (ID_i, x_i, P_i) \) to \( K^{\text{list}} \).

Without loss of generality, we assume that before \( A \) makes the following queries, he has already submitted the related identities to the Public-Key-Oracle.

Secret-Key-Oracle: For input \( ID_i, C \) first finds the tuple \( (ID_i, x_i, P_i) \) on \( K^{\text{list}} \), then returns \( x_i \) as answer.

Public-Key-Replacement-Oracle: For input \( (ID_i, P_i) \), \( C \) first finds the tuple \( (ID_i, x_i, P_i) \) on \( K^{\text{list}} \), then \( C \) sets \( x_i = \bot \) and \( P_i = P_i' \).

Sign-Oracle: On input \( (ID_i, P_i, M_i) \), \( C \) first makes the corresponding hash queries which will be used then recover \( (ID_i, \mu_i, Q_i, D_i, c_i) \), \( (M_i, ID_i, P_i, \pi_i, U_i) \) from \( H_1^{\text{list}} \) and \( H_3^{\text{list}} \) respectively, and does the following:

1. If \( c_i = 0 \), choose \( R_i \in G_1 \), submit \( (M_i, ID_i, P_i, R_i) \) to \( H_2 \)-Oracle and find \( (M_i, ID_i, P_i, R_i, \nu_i, F_i) \) on \( H_2^{\text{list}} \) later. Finally, set \( V_i = \mu_i P_T + \nu_i R_i + \pi_i P \) and return \( (R_i, V_i) \) as answer.

2. Else, randomly choose \( r_i, \nu_i \in Z_q^* \), set \( R_i = r_i P - \nu_i^{-1} Q_i \), submit \( (M_i, ID_i, P_i, R_i) \) to \( H_2 \)-Oracle. If \( (M_i, ID_i, P_i, R_i) \) has submitted to \( H_2 \)-Oracle previously, redo Step 2. Otherwise, define \( F_i = \nu_i P \), add \( (M_i, ID_i, P_i, R_i, \nu_i, F_i) \) to \( H_2^{\text{list}} \) and set \( V_i = r_i F_i + \pi_i P \). Finally, return \( (R_i, V_i) \) as answer.

Forgery stage: At the end of \( A \)’s attack, he outputs a tuple \( \{L_M = \{M_1, \ldots, M_n\}, L^{\text{ID}}_M = \{ID_1, \ldots, ID_n\}, L^{\text{PK}}_M = \{P_1, \ldots, P_n\}, \sigma^* = (R_1, \ldots, R_n, V^*) \} \). For all \( i \in \{1, \ldots, n\} \), \( C \) recovers \( (ID_i, \mu_i, Q_i, D_i, c_i) \).
\( R_i, \nu_i, F_i, (M_i, ID_i, P_i, \pi_i, U_i) \) from \( H_1^{ist}, H_2^{ist} \) and \( H_3^{ist} \) respectively. We say \( \sigma^* \) is a valid and nontrivial forgery if all of the following conditions are satisfied: \( \sigma^* \) could pass the verification algorithm \textsf{Aggregate Verify}, one of \( c_i^* = 1 \) (without loss of generality, we assume \( c_1^* = 1 \)) and \( (ID_i^*, P_i^*, M_i^*) \) has not submitted to the Sign-Oracle. If \( \sigma^* \) is a valid and nontrivial forgery, \( \mathcal{C} \) outputs \( abP = \mu_1^{-1}(V - \Sigma_{i=1}^n(\nu_i^* R_i^* - \pi_i^* P_i^*) - \Sigma_{i=2}^n(\mu_i^* P_T)) \) as the solution of the CDH problem. Otherwise, \( \mathcal{C} \) will abort.

To complete the proof, we shall show that \( \mathcal{C} \) solves the given instance of CDH problem with probability at least \( \epsilon' \).

First, we analyze the three events needed for \( \mathcal{C} \) to succeed:

- \( \Gamma_1: \mathcal{C} \) does not abort as a result of any of \( \mathcal{A} \)'s queries.
- \( \Gamma_2: \mathcal{A} \) generates a valid and nontrivial forgery.
- \( \Gamma_3: \Gamma_2 \) occurs, and \( c_i^* = 1 \) and \( c_i = 0 \) for \( 2 \leq i \leq n \).

\( \mathcal{C} \) succeeds if all of these events happen. The probability \( Pr[\Gamma_1 \land \Gamma_2 \land \Gamma_3] \) can be decomposed as \( Pr[\Gamma_1 \land \Gamma_2 \land \Gamma_3] = Pr[\Gamma_1]Pr[\Gamma_2 \mid \Gamma_1]Pr[\Gamma_3 \mid \Gamma_1 \land \Gamma_2] \).

It is easy to have \( \epsilon' = Pr[\Gamma_1 \land \Gamma_2 \land \Gamma_3] \geq (1 - \delta)q_p^{n+1} \epsilon \delta \). Then the probability turns to \( \frac{1}{c(q_p + \eta_2)^2} \). And \( \tau' = \tau + \Theta(q_T \tau G_1) \), where \( \tau G_1 \) is the time to compute a multiplication in \( G_1 \).

**Theorem 2**: Our CLAS scheme is unforgeable against a super type II adversary in the random oracle model assuming the CDH problem in \( G_1 \) is intractable.

The proof goes very similar to our Theorem 1 and Lemma 2 in [14] and hence, it is omitted.

## 5 Conclusion

Certificateless public key cryptography effectively solves the inherent key escrow problem in ID-PKC while keeps its certificate free property. In this paper, we have shown the drawbacks of the security model in [5]. Moreover, we have presented a new security model as well as a certificateless aggregate signature scheme. In our security model, we treat type I and type II adversaries as super adversaries. Hence, our model commendably captures the most powerful attack of type I and type II adversaries.

## Acknowledgment

Project supported by the National Natural Science Foundation of China (No. 60673070), the Natural Science Foundation of Jiangsu Province (No. BK2006217).

## References


