On the Security of MTA-OTIBASs (Multiple-TA One-Time Identity-Based Aggregate Signatures)

Lei Zhang, Member, IEEE, Qianhong Wu, Member, IEEE, Josep Domingo-Ferrer, Fellow, IEEE
Bo Qin, Chuanyan Hu

Abstract—In [3] the authors proposed a new aggregate signature scheme referred to as multiple-TA (trusted authority) one-time identity-based aggregate signature (MTA-OTIBAS). Further, they gave a concrete MTA-OTIBAS scheme. We recall here the definition of MTA-OTIBAS and the concrete proposed scheme. Then we prove that our MTA-OTIBAS concrete scheme is existentially unforgeable against adaptively chosen-message attacks in the random oracle model under the co-CDH problem assumption.

Index Terms—Identity based cryptosystem, Signature, Aggregate signature

1 INTRODUCTION
In [3] we proposed a new aggregate signature scheme referred to as multiple-TA (trusted authority) one-time identity-based aggregate signature (MTA-OTIBAS). Further, we gave a concrete MTA-OTIBAS scheme. We first recall the notion of MTA-OTIBAS; we then recall its formal definition and the concrete scheme proposed in [3]. Then, we give the detailed security proof of MTA-OTIBAS (not given in [3]).

An MTA-OTIBAS scheme has the following features. Firstly, each user’s public key is his identity, so no certificate is needed on the public key, which avoids the certificate management overhead. Secondly, a signer’s private key (corresponding to an identity and a lower-level TA) is restricted to be used only once; after that, the signer’s private key should be updated. Thirdly, the MTA-OTIBAS scheme also allows signature aggregation and fast verification, i.e., $n$ signatures can be aggregated into a single short signature (even signatures generated by signers enrolled by different lower-level TAs), which greatly saves storage space, and can be verified simultaneously.

We recall the formal definition of MTA-OTIBAS in Section 2. In Section 3 we recall the concrete MTA-OTIBAS scheme. Then in Section 4 we prove that our MTA-OTIBAS concrete scheme is existentially unforgeable against adaptively chosen-message attacks in the random oracle model under the co-CDH problem assumption.

2 DEFINITION OF MTA-OTIBAS
An MTA-OTIBAS scheme consists of six algorithms, i.e., Root.Setup, LowLevel.Setup, Extract, Sign, Aggregate, and Verify. Root.Setup is run by the root TA to generate the global system parameters and system master key. LowLevel.Setup is an interactive protocol run between a lower-level TA and the root TA. It generates the secret key, public key and certificate of the lower-level TA. Extract takes as input a lower-level TA’s secret key and a signer’s identity, and outputs a private key for the signer. Sign takes as input a signer’s identity, his private key, the certificate of the signer’s corresponding lower-level TA and any message, and outputs a signature on the message. The signature is only valid under the signer’s identity and the certificate of his corresponding lower-level TA. A restriction here is that a private key corresponding to a specific identity issued by a lower-level TA can be used only once. However, the same identity can be enrolled by different lower-level TAs. This implies that the corruption of a lower-level TA does not influence the signers enrolled by other lower-level TAs. Aggregate is used to aggregate $n$ message-signature pairs generated by the Sign procedure into a single signature, i.e., an aggregate signature. Verify is used to check the validity of an aggregate signature. It takes as input $n$ messages, the corresponding aggregate signature, $n$ identities enrolled by $l$ lower-level TAs, and outputs 1 or 0 to represent whether the aggregate signature is valid or not.

3 A CONCRETE MTA-OTIBAS SCHEME
Our MTA-OTIBAS scheme is realized using bilinear maps which are widely employed in identity-based cryptosystems. A map $e : G_1 \times G_2 \rightarrow G_T$ is called a bilinear map if $e(g_1, g_2) \neq 1$ and $e(g_1^\alpha, g_2^\beta) = e(g_1, g_2)^{\alpha\beta}$ for all $\alpha, \beta \in \mathbb{Z}_q^*$, where $G_1, G_2$ are two cyclic groups of prime order $q$, $G_T$ is a multiplicative cyclic group of the same order, $g_1$ is a generator of $G_1$, and $g_2$ is a generator of $G_2$. By exploiting bilinear maps, we implement our MTA-OTIBAS scheme.

Root.Setup: The root TA runs this algorithm to generate
the system parameters as follows:

1. Choose \( q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \hat{e}, \psi \), where \( \psi \) is a computable isomorphism from \( \mathbb{G}_2 \) to \( \mathbb{G}_1 \), with \( \psi(g_2) = g_1 \).
2. Pick \( k \in \mathbb{Z}_q^* \) as its master secret key, and compute \( y = g_2^k \) as its public master key.
3. Select cryptographic hash functions \( H_0(\cdot) : \{0, 1\}^* \rightarrow \mathbb{G}_2 \) and \( H_1(\cdot) : \{0, 1\}^* \rightarrow \mathbb{Z}_q^* \).
4. Publish the system global parameter \( \Psi = (\hat{e}, q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, H_0(\cdot), H_1(\cdot), \psi) \).

**LowerLevel.Setup:** In an MTA-OTIBAS scheme, before a lower-level TA can recruit members, it must be enrolled by the root TA. The root TA may add the public information of a lower-level TA (e.g., identity and public key) to the system global parameters. Let the identity of a lower-level TA \( T_i \) be \( I_D T_i \). \( T_i \) picks \( \kappa_i \in \mathbb{Z}_q^* \) as its secret key and computes \( y_i = g_2^{\kappa_i} \) as its public key. \( (I_D T_i, y_i) \) are submitted to the root TA. On input \( (I_D T_i, y_i) \), the root TA generates a certificate \( cert_{T_i} \), which is signed using its master secret key. Finally, \( cert_{T_i} \) is sent to \( T_i \).

**Extract:** Suppose a signer with identity \( I_D j \) wants to join the system maintained by \( T_i \), whose secret key is \( \kappa_i \). On input the signer’s identity \( I_D j \), \( T_i \) generates the private key for the signer as follows:

1. Compute \( id_{j,0} = H_0(I_D j, 0), id_{j,1} = H_0(I_D j, 1) \);
2. Compute \( s_{j,i} = id_{j,0}^{\kappa_i}, s_{j,1} = id_{j,1}^{\kappa_i} \), and set \( s_{j,i} = (s_{j,0,i}, s_{j,1,i}) \) as the private key of the signer.

**Sign:** To sign a message \( m_k \), a signer with identity \( I_D j \) enrolled by \( T_i \) and private key \( s_{j,0,i}, s_{j,1,i} \) computes \( h_k = H_1(m_k, I_D j, cert_{T_i}), \sigma_k = s_{j,0,i}s_{j,1,i} \). The signer outputs \( \sigma_k \) as the signature on \( m_k \).

**Aggregate:** This publicly computable algorithm aggregates \( n \) signatures into a single signature. Let an entity collect \( n \) message-signature pairs \( \{(m_1, \sigma_1), \ldots, (m_n, \sigma_n)\} \) signed by \( n \) users with corresponding identities \( \{I_D 1, \ldots, I_D n\} \) enrolled by \( l \) lower-level TAs \( \{T_1, \ldots, T_l\} \). For simplicity, we assume \( \{I_D 1, \ldots, I_D l\}, \{I_D l+1, \ldots, I_D l+2\}, \ldots, \{I_D l+n-1, \ldots, I_D l+n\} \) are enrolled by \( T_{l+1}, \ldots, T_l \) respectively. The message-signature pairs are divided into \( l \) sets corresponding to the \( l \) lower-level TAs. This algorithm outputs \( \Omega \) as the resulting aggregate signature, where \( \Omega = \prod_{i=1}^n \sigma_i \).

**Verify:** To verify a signature aggregate \( \Omega \) on messages \( \{m_1, \ldots, m_n\} \) under \( \Pi = \{I_D 1, \ldots, I_D l\}, \Pi_2 = \{I_D l+1, \ldots, I_D l+2\}, \ldots, \Pi_n = \{I_D l+n-1, \ldots, I_D l+n\} \) enrolled by \( T_{l+1}, \ldots, T_l \) respectively, the verifier performs the following steps:

1. For \( 1 \leq j \leq n \), compute \( h_j = H_1(m_j, I_D j, cert_{T_i}) \) and \( id_{j,0} = H_0(I_D j, 0), id_{j,1} = H_0(I_D j, 1) \).
2. Define \( \Pi_i = \{1, \ldots, t_i\}, \Pi_i = \{t_i + 1, \ldots, t_{i+1}\}, \ldots, \Pi_i = \{t_{n-1} + 1, \ldots, n\} \). Check \( \hat{e}(\Omega, g_2) = \prod_{i=1}^n \hat{e}(\Pi_i, id_{j,0}^{\kappa_i}, id_{j,1}) \). Output 1 if the equation holds; else output 0.

### 4 Security Proof

An MTA-OTIBAS scheme should be secure. Informally, an MTA-OTIBAS scheme is said to be secure if no polynomial-time attacker not requesting a private key of an entity enrolled by a lower-level TA can forge an aggregate signature that is valid (i.e., such that \( \text{Verify} \) outputs 1) corresponding to that entity enrolled by the lower-level TA.

In general, the security of an MTA-OTIBAS scheme is modeled via the following EUF-CMA (existential universal forgery under adaptive chosen-message attack) game \( \Pi \) and takes place between a challenger \( CH \) and an adversary \( A \). The game has the following three stages:

**Initialize:** \( CH \) runs the Root.Setup algorithm to obtain a master secret key and the system parameters. \( CH \) then sends the system parameters to \( A \) while keeping secret the master secret key.

**Attack:** \( A \) can perform a polynomially bounded number of the following types of queries in an adaptive manner.

- **LowerLevel.Setup queries:** \( A \) may ask \( CH \) to set up a lower-level TA. On input an identity \( I_D T_i \) of a lower-level TA, \( CH \) generates the secret key and certificate of the lower-level TA.
- **Corrupt.LowerLevel queries:** \( A \) can request the secret key of a lower-level TA \( T_i \). On input \( I_D T_i, CH \) outputs the corresponding secret key of \( T_i \).
- **Extract queries:** \( A \) can request the private key of an entity with identity \( I_D j \) issued by a lower-level TA \( T_i \). On input \( I_D j, cert_{T_i} \), \( CH \) outputs the corresponding private key of the entity.
- **Sign queries:** \( A \) can request an entity’s signature on a message \( m_k \). On receiving a query on \( (m_k, I_D j, cert_{T_i}) \), \( CH \) generates a valid signature \( \sigma_j \) on \( m_k \) under \( (I_D j, cert_{T_i}) \), and replies with \( \sigma_j \).

**Forgery:** \( A \) outputs \( l' \) sets of identities \( \Pi'_1 = \{I_D 1, \ldots, I_D l'_1\}, \Pi'_2 = \{I_D l'_1 + 1, \ldots, I_D l'_2\}, \ldots, \Pi'_{l'} = \{I_D l'_l - 1 + 1, \ldots, I_D l'_l\} \) enrolled by \( l' \) lower-level TAs with certificates from the set \( \{cert_{T_{l+1}}, \ldots, cert_{T_l}\} \), a set of \( n \) messages \( \{m_1, \ldots, m_n\} \) and an aggregate signature \( \sigma^* \). For simplicity, we assume \( m_i \) corresponds to \( I_D i \) for \( i \in \{1, \ldots, n\} \).

\( A \) wins the above game, if all of the following conditions are satisfied:

1. \( \sigma^* \) is a valid aggregate signature on messages \( \{m_1, \ldots, m_n\} \) under \( \Pi'_1 = \{I_D 1, \ldots, I_D l'_1\}, \Pi'_2 = \{I_D l'_1 + 1, \ldots, I_D l'_2\}, \ldots, \Pi'_l = \{I_D l'_l - 1 + 1, \ldots, I_D l'_l\} \) and \( \{cert_{T_{l+1}}, \ldots, cert_{T_l}\} \).
2. At least, one private key of an entity issued by a lower-level TA is not queried by \( A \) during the Extract queries and the lower-level TA is not corrupted. Without loss of generality, we assume the identity of the entity is \( I_D i \) and its corresponding lower-level TA is \( T_i^* \) with certificate \( cert_{T_i^*} \).
3. For a message \( m \neq m_i \), the query \( (m, I_D i, cert_{T_i}) \) can be queried at most once, and \( (m_i, I_D i, cert_{T_i}) \) is never queried during the Sign queries.

We can now define the security of an MTA-OTIBAS scheme in terms of the above game.
**Definition 1.** An MTA-OTIBAS scheme is secure, i.e., secure against existential forgery under adaptive chosen-message attack, if the success probability of any polynomially bounded adversary in the above EUF-CMA game is negligible.

We next recall the co-CDH assumption on which the security of the signature scheme in Section 4 rests.

**Definition 2 (co-CDH Assumption).** The co-CDH assumption in two cyclic groups $G_1$ and $G_2$ of prime order $q$ equipped with bilinearity states that, given $(g_1^a, g_2^b)$ for randomly chosen $a, b \in \mathbb{Z}_q^*$, it is hard for any polynomial-time algorithm to compute $g_1^{ab}$.

Regarding the security of our MTA-OTIBAS scheme, we have the following claim.

**Theorem 1.** Assume an adversary $A$ has an advantage $\epsilon$ in forging an MTA-OTIBAS scheme of Section 3 in an attack modeled by the above EUF-OTIBAS-CMA game, within a time span $\hat{\tau}$; the adversary can make at most $q_{u_1}$ times $H_1(\cdot)$ ($i = 0, 1$) queries, $q_s$ times LowerLevel.Setup queries, $q_c$ times Corrupt.LowerLevel queries, $q_e$ times Extract queries, $q_s$ times Sign queries. Then the challenger can solve the co-CDH problem with probability $\epsilon' \geq \epsilon^2 \left( 4q_c + q_e + q_s + n + 2\hat{\tau} \right) \epsilon$ within time $\hat{\tau}' = \hat{\tau} + O(q_{u_1} + q_s + q_e + n + 2\hat{\tau} \epsilon)$, where $\epsilon$ is the time to compute a point exponentiation in $G_1$ and $n$ is the size of the aggregating set.

**Proof:** Let $C\mathcal{H}$ be a co-CDH attacker who receives a co-CDH challenge instance $(g_1^a, g_2^b)$ and wants to compute the value of $g_1^{ab}$. $A$ is an adversary who interacts with $C\mathcal{H}$ as modeled in the EUF-CMA game. We show how $C$ can use $A$ to break the co-CDH assumption.

**Initialize:** Firstly, $C\mathcal{H}$ selects $\Psi = (\hat{e}, q, G_1, G_2, G_T, g_1, g_2, y, H_0(\cdot), H_1(\cdot), \hat{\psi})$, where $y = g_2^b$, and $\kappa$ is the master secret key; then $\Psi$ is sent to $A$.

**Attack:** We consider the hash functions $H_0(\cdot)$ and $H_1(\cdot)$ as random oracles. $A$ can perform the following types of queries in an adaptive manner.

**$H_0(\cdot)$ queries:** $C\mathcal{H}$ maintains a list $H_0^{list}$ of tuples $(ID_i, \alpha_0, \alpha_1, \alpha'_0, \alpha'_1, \epsilon_0, \epsilon_1, id_i, 0, id_i, 1, \text{embed}_i)$. This list is initially empty. Whenever $C\mathcal{H}$ receives an $H_1$ query on $(ID_i, j)$ (where $j = 0$ or 1), $C\mathcal{H}$ does the following:

- If $ID_i$ exists in a previous query, find $(ID_i, \alpha_0, \alpha'_0, \alpha_1, \alpha'_1, \epsilon_0, \epsilon_1, id_i, 0, id_i, 1, \text{embed}_i)$ on $H_0^{list}$ and return $id_i, j$.

- Else, first flip a coin $\text{coin}_i \in \{0, 1\}$ that yields 1 with probability $\delta$ and 0 with probability $1 - \delta$. Then do:
  - If $\text{coin}_i = 0$, select $\alpha_0, \alpha'_0, \alpha_1, \alpha'_1 \in \mathbb{Z}_q^*$, compute $id_i, 0 = \alpha_0, \alpha'_0, \alpha_1, \alpha'_1, \epsilon_0, \epsilon_1, id_i, 0, id_i, 1, \text{embed}_i)$ to $H_0^{list}$.
  - Else randomly select $\alpha_0, \alpha'_0, \alpha_1, \alpha'_1 \in \mathbb{Z}_q^*$, set $id_i, 0 = \alpha_0, \alpha'_0, \alpha_1, \alpha'_1, \epsilon_0, \epsilon_1, id_i, 0, id_i, 1, \text{embed}_i)$ to $H_0^{list}$.

**LowerLevel.Setup queries:** $C\mathcal{H}$ maintains a list $T_{A list}$ of tuples $(ID_T, \kappa, y_i, \text{cert}_T, \text{coin}_T)$. On input an identity $ID_T$ of a lower-level TA, $C\mathcal{H}$ does the following:

- If there is a tuple $(ID_T, \kappa, y_i, \text{cert}_T, \text{coin}_T)$ on $T_{A list}$, return $\text{cert}_T$ as the answer.

- Else, choose $\kappa_i \in \mathbb{Z}_q^*$, flip a coin $\text{coin}_T \in \{0, 1\}$ that yields 1 with probability $\delta$ and 0 with probability $1 - \delta$ and do the following:
  - If $\text{coin}_T = 0$, set $\kappa_i$ as the secret key, compute $y_i = g_0^{\kappa_i}$, generate a certificate $\text{cert}_T$ corresponding to $(ID_T, y_i)$, add $(ID_T, \kappa, y_i, \text{cert}_T, \text{coin}_T)$ to $T_{A list}$.
  - Else, compute $y_i = g_2^{b\kappa}$, generate a certificate $\text{cert}_T$ corresponding to $(ID_T, y_i)$, add $(ID_T, \kappa, y_i, \text{cert}_T, \text{coin}_T)$ to $T_{A list}$.

In the rest of this paper, we assume that if a certificate $\text{coin}_T$ appears, $A$ has already made a corresponding LowerLevel.Setup query.

**$H_1(\cdot)$ queries:** $C\mathcal{H}$ keeps a list $H_1^{list}$ of tuples $(ID_i, m_i, \text{cert}_i, y_i, \text{coin}_i)$. This list is initially empty. Whenever $A$ issues a query $H_1(ID_i, m_i, \text{cert}_T, y_i)$, $C\mathcal{H}$ does the following:

- If there is a tuple $(ID_i, m_i, \text{cert}_T, y_i, \text{coin}_i)$ on $H_1^{list}$, return $h_i$ as the answer.

- Else, submit $(ID_i, 0)$ to $H_0$ and recover the tuple $(ID_i, \alpha_0, \alpha'_0, \alpha_1, \alpha'_1, id_i, 0, id_i, 1, \text{coin}_i)$ from $H_0^{list}$, recover the tuple $(ID_T, \kappa, y_i, \text{cert}_T, \text{coin}_T)$ from $T_{A list}$, flip a coin $\text{coin}_i \in \{0, 1\}$ that yields 1 with probability $\delta$ and 0 with probability $1 - \delta$. Then do the following:
  - If $\text{coin}_i = \text{coin}_T = 1$, and $\text{coin}_T = 1$, add $(ID_i, m_i, \text{cert}_T, y_i, \text{coin}_i)$ to $H_1^{list}$ and return $h_i = -\alpha_0/\alpha_1$ as the answer.

  - Else, randomly select $h_i \in \mathbb{Z}_q^*$, add $(ID_i, m_i, \text{cert}_T, y_i, \text{coin}_i)$ to $H_1^{list}$ and return $h_i$ as the answer.

**Corrupt.LowerLevel queries:** On input an identity $ID_T$ of a lower-level TA, $C\mathcal{H}$ first makes a LowerLevel.Setup query on $ID_T$, and recovers the tuple $(ID_T, \kappa, y_i, \text{cert}_T, \text{coin}_T)$ on $T_{A list}$. If $\text{coin}_T = 0$, $C\mathcal{H}$ returns $\kappa_i$ as the answer; otherwise, $C$ aborts.

**Extract queries:** When $A$ issues an Extract query on $(ID_i, \text{cert}_T)$, the same answer will be given if the request has been asked before. Otherwise, $C\mathcal{H}$ recovers $(ID_T, \kappa, y_i, \text{cert}_T, \text{coin}_T)$ from $T_{A list}$; $C$ checks whether $(ID_i, \alpha_0, \alpha'_0, \alpha_1, \alpha'_1, id_i, 0, id_i, 1, \text{coin}_i)$ is on $H_0^{list}$; if it is not, $C\mathcal{H}$ submits $(ID_i, j)$ to $H_0(\cdot)$ to generate such a tuple, where $j = 0$ or 1. Finally, if $\text{coin}_T = \text{coin}_T = 1$, $C\mathcal{H}$ aborts; else if $\text{coin}_T = 0$, it returns $(id_i, 0, id_i, 1)$. Otherwise, $C\mathcal{H}$ returns $(\hat{\psi}(y_2^{b\kappa}, \alpha_0, \alpha'_0, \alpha_1, \alpha'_1, id_i, 0, id_i, 1, \text{coin}_i))$.

**Sign query:** On receiving a Sign query on $(ID_i, m_i, \text{cert}_T)$, $C\mathcal{H}$ first queries $H_0(ID_i, j)$ ($j = 0$ or 1), LowerLevel.Setup($ID_T$) and $H_1(ID_i, m_i, \text{cert}_T)$ if they were not queried before, then recovers $(ID_i, \alpha_0, \alpha'_0, \alpha_1, \alpha'_1, id_i, 0, id_i, 1, \text{coin}_i)$ from $H_0^{list}$, $(ID_T, \kappa, y_i, \text{cert}_T, \text{coin}_T)$ from $T_{A list}$ and $(ID_i, m_i, \text{cert}_T, y_i, \text{coin}_i)$ from $H_1^{list}$. Finally $C\mathcal{H}$ generates the signature as follows:

- If $\text{coin}_i = \text{coin}_T = \text{coin}_T = 1$, compute and output $\sigma_i = \hat{\psi}(y_2^{b\kappa}, \alpha_0, \alpha'_0, \alpha_1, \alpha'_1, id_i, 0, id_i, 1, \text{coin}_i))$. 


Claim 1. The probability that $\mathcal{H}$ does not abort is

$$\Pr[\Sigma_1^{\text{Corrupt.LowerLevel}}] = 1.$$ 

Proof: For a Corrupt.LowerLevel query, $\mathcal{H}$ will abort iff $\text{coin}_i = \text{coin}_T = 1$. It is easy to see that the probability that $\mathcal{H}$ does not abort as a result of $\mathcal{A}$'s Corrupt.LowerLevel queries is at least $(1 - \delta)^{q_{\mathcal{C}}}$. 

For an Extract query, $\mathcal{H}$ will abort iff $\text{coin}_i = \text{coin}_T = 1$. It is easy to see that the probability that $\mathcal{H}$ does not abort for an Extract query is $1 - \delta^2 > 1 - \delta$. Since $\mathcal{A}$ can make at most $q_{\mathcal{C}}$ times Extract queries, the probability that $\mathcal{H}$ does not abort as a result of $\mathcal{A}$'s Extract queries is at least $(1 - \delta)^{q_{\mathcal{C}}}$. 

When $\mathcal{H}$ receives a Sign query, he will abort iff $\text{coin}_i = \text{coin}_T = 1$, $\text{coin}_i' = 0$ happen. So for a Sign query, the probability that $\mathcal{H}$ does not abort is $1 - \delta^2(1 - \delta) > 1 - \delta$. Since $\mathcal{A}$ makes at most $q_{\mathcal{C}}$ times Sign queries, the probability that $\mathcal{H}$ does not abort as a result of $\mathcal{A}$'s Sign queries is at least $(1 - \delta)^{q_{\mathcal{C}}}$. 

Overall, we have $\Pr[\Sigma_1] > (1 - \delta)^{q_{\mathcal{C}} + q_{\mathcal{E}} + q_{\mathcal{S}}}$. 

Claim 2. $\Pr[\Sigma_2|\Sigma_1] \geq \epsilon$.

Proof: If $\mathcal{H}$ does not abort, then $\mathcal{A}$'s view is identical to its view in the real attack. Hence, $\Pr[\Sigma_2|\Sigma_1] \geq \epsilon$. 

Claim 3. The probability that $\mathcal{H}$ does not abort after $\mathcal{A}$ outputting a valid and nontrivial forgery is at least $\delta(1 - \delta)^n$. Hence $\Pr[\Sigma_3|\Sigma_1 \land \Sigma_2] \geq \delta(1 - \delta)^n$. 

Proof: Events $\Sigma_1$ and $\Sigma_2$ have occurred, and $\mathcal{A}$ has generated a valid and nontrivial forgery $(ID_1^*, ..., ID_n^*, m_1^*, ..., m_n^*, \Omega^*)$. $\mathcal{H}$ will abort unless $\mathcal{A}$ generates a forgery such that there exists an $i \in \{1, ..., n\}$ such that $\text{coin}_{i}^1 = \text{coin}_{T, i}^1 = 1$, $\text{coin}_{i}^1 = 0$, and for $2 \leq i \leq n$, $\text{coin}_{i}^1 = 0$. Therefore, $\Pr[\Sigma_3|\Sigma_1 \land \Sigma_2] \geq \delta(1 - \delta)^n$. 

In total, we have $\epsilon' = \Pr[\Sigma_1 \land \Sigma_2 \land \Sigma_3] > (1 - \delta)^{q_{\mathcal{C}} + q_{\mathcal{E}} + q_{\mathcal{S}}}(1 - \delta)^n \epsilon \geq \frac{4}{e(4(4q_{\mathcal{C}} + 4q_{\mathcal{E}} + 4q_{\mathcal{S}} + 4))}$, where $e$ is Euler's constant.

5 Conclusion

We have proven that our MTA-OTIBAS concrete scheme is existentially unforgeable against adaptively chosen-message attacks in the random oracle model under the co-CDH problem assumption.

References

[1] C. Gentry and Z. Ramzan, “Identity-based aggregate signatures,” in Proc. PKC 2006, pp. 257-273.
[2] L. Zhang, Q. Wu, A. Solanas and J. Domingo-Ferrer, “A scalable robust authentication protocol for secure vehicular communications,” IEEE Trans. Veh. Technol., vol. 59, no. 4, pp. 1606-1617, 2010.
[3] L. Zhang, Q. Wu, J. Domingo-Ferrer, B. Qin and C. Hu, “Distributed aggregate privacy-preserving authentication in VANETs,” manuscript, 2015.