International Journal of Information & Network Security (IJINS)
Vol.3, No.6, December 2014, pp. 370~380
ISSN: 2089-3299

Efficient Aggregate Proxy Signature Scheme in ID-based Framework

P.V.S.S.N. Gopal*, P. Vasudeva Reddy*, T. Gowri**
* Department of Engineering Mathematics, Andhra University
** Department of Electronics and Communication Engineering, GITAM University

Article Info

Received Jun 12th, 2014
Revised Aug 20th, 2014
Accepted Aug 26th, 2014

Keyword:

ID-based Cryptography
Aggregate Proxy Signature
Bilinear Pairings
Random Oracle Paradigm
CDH Problem

ABSTRACT

Ever since Mambo et al. [1] and Boneh et al. [2] introduced the notions of proxy signature and aggregate signature in 1996 and 2003 respectively; the cryptographic research took a rapid progress in these areas. Proxy signatures play a vital role in many real world applications when signatures are to be generated in the absence of the original signer. Aggregate signature schemes have wider applications and dramatically reduce the communication bandwidth and computational overhead. Keeping the merits of proxy and aggregate signatures, in this work, we propose an efficient aggregate proxy signature in Identity-based framework using bilinear pairings. This scheme achieves constant signature size and constant pairings operations for aggregate verification. We prove the security of the proposed scheme in random oracle paradigm, tightly related to the Computational Diffie-Hellman (CDH) problem. We compare the proposed scheme with related schemes.

Corresponding Author:

P. Vasudeva Reddy,
Department of Engineering Mathematics,
Andhra University,
Visakhapatnam-530003,
Andhra Pradesh, INDIA.
Email: vasucrypto@yahoo.com

1. INTRODUCTION

To overcome the task of maintaining certificate libraries used for revoking, storage and distribution of certificates which require huge communication overload in Public Key Infrastructure (PKI) based setting, Shamir [3] in 1984, devised the paradigm called Identity Based Cryptosystem (IBC). In this system the public key of a user can be directly derived from his/her personal identity like telephone number, e-mail address etc. and the corresponding private key is issued by a trusted authority termed Key Generation Centre (KGC). Later on, many encryption and signature schemes have been constructed in IBC setting, but the most usable and practical encryption scheme using Weil pairing was devised by Boneh et al. [4], in 2001. Based on the work in [4], many signature schemes in the ID based setting were proposed in the literature [5-8].

The concept of aggregate signature was introduced by Boneh et al. [2] in 2003. In this scheme a single compressed signature is obtained upon combining different *n* signatures from different *n* users on different *n* messages. Such signature can be verified by anyone and convince himself/herself that the *n* user's undeniably signed the *n* original messages. Certainly, the performance of a signature scheme can be calculated using computational overhead, but reducing communication bandwidth i.e. the signature size is equally important. Aggregate signature is one such approach towards achieving this task. Based on the work in [2], many aggregate signature scheme appeared in the literature in ID-based setting [9-14].

The concept of proxy signature is introduced by Mambo et al. [1] in 1996. In this scheme the original signer delegates his/her signing capability using a warrant consisting of delegation rights to a proxy signer. Anyone can verify the validity of the proxy signature using the warrant delegated by the original signer. Such schemes have many real world applications, such as the director of a company in his/her absence delegates his/her signing rights to the concerned managers. Moreover, aggregating the proxy signatures signed by the proxy signers and resulting into a single compact signature enables a verifier with less computational overhead and reduced communication bandwidth. Based on the work in [1], many proxy signature schemes in the ID-based setting appeared in the literature [15-18].

In 2013, Lin et al. [19] proposed an ID-based aggregate proxy signature scheme realizing warrant-based delegation. This scheme requires 3 pairing operations in the aggregate signature verification phase and its security reduction is obtained using Forking lemma [20].

To meet the demands of aggregate and proxy signatures, in this paper, we proposed an ID-based Aggregate Proxy Signature (IBAPS) scheme, which requires only 2 (constant) pairing operations in the aggregate verification phase and of constant size, irrespective of the number of proxy signers participate in signing. We proved the security model of our scheme in the random oracle model under CDH assumption without using Forking lemma and hence the obtained security reduction is tight.

Rest of the paper is organized as follows: Section 2 gives some preliminaries, including bilinear maps and complexity assumptions. In Section 3, syntax and security model of IDAPS scheme is presented. Our efficient IDAPS scheme is presented in Section 4. Security of the proposed scheme is proved in Section 5. In Section 6, we compare our scheme with the related schemes. Finally, Section 7 concludes our work.

2. PRELIMINARIES

This section summarizes some fundamental concepts and necessary hard problems related to our scheme.

2.1 Bilinear Map

Let G and G_T are cyclic groups under addition and multiplication respectively, both of same prime order q with P as a generator in $G.$ A map $\hat{e}: G \times G \rightarrow G_T$ is called bilinear if the following properties are satisfied:

1. **Bilinear:** $\forall\ A,\ B \in G,\ \forall\ x,\ y \in Z_q^*,\ \ \hat{e}(xA,\ yB) = \hat{e}(A,\ B)^{xy}.$
2. **Non-Degeneracy:** $\exists\ A \in G,\ \ni\ \hat{e}(A,\ A) \neq 1.$
3. **Computable:** $\forall\ A,\ B \in G,\ \ \hat{e}(A,\ B)$ can be computable using an efficient algorithm.

Upon making suitable variations in the Weil or Tate pairing one can obtain such maps on elliptic curves over a finite field [4, 21].

2.2 Complexity Assumptions

In the following, we present some necessary hard problems on which the proposed scheme's security is based.

- **Computational Diffie-Hellman (CDH) Problem:** $\forall\ x,\ y \in Z_q^*,$ given $P,\ xP,\ yP \in G$ evaluate $xyP \in G.$
- **Decision Diffie-Hellman (DDH) Problem:** $\forall\ x,\ y,\ z \in Z_q^*,$ given $P,\ xP,\ yP,\ zP \in G$ decide whether $z = xy.$ If so, the tuple $(P,\ xP,\ yP,\ zP)$ is called a valid Diffie-Hellman tuple.

It is believed; in general that solving CDH problem with non negligible advantage cannot be done in polynomial time.

- **Gap Diffie-Hellman (GDH) Group:** A group G is said to be a GDH group if there is a probabilistic polynomial time algorithm to evaluate the DDH problem but such algorithm do not exist to evaluate the CDH problem.

3. SYNTAX AND SECURITY MODEL OF THE PROPOSED IDAPS SCHEME

In this section we present the syntax and security model of our proposed scheme.

3.1 Syntax of IDAPS scheme

An IDAPS scheme involves a KGC, an original signer P_0, an aggregating set *L* of *n* proxy users/signers $P_1, P_2, ..., P_n$ and an aggregate proxy signature generator. The proposed IDAPS scheme comprises six polynomial time algorithms: System Setup, Key Extraction, Warrant Delegation, Proxy Signature Generation, Aggregation and Aggregate Proxy Signature Verification. Detailed functionalities of these algorithms are presented below.

System Setup: For a given security parameter *l*, the KGC outputs the system parameters *Params* and the master private key <*s*>. *Params* are made public, where as <*s*> is kept secret. *Params* are the necessary input for the remaining algorithms.

Key Extraction: This algorithm run by the KGC takes as input the *Params*, identity ID_i of a signer P_i $(i = 0, 1, 2, ..., n)$; outputs the private key for ID_i and forwards it to the corresponding user over a secure channel.

Warrant Delegation: In this, the original signer P_0 delegates his signing power to the proxy signers P_i $(i = 1, 2, ..., n)$; by sending his/her signature for a warrant *w* to each P_i. *w* consists of all the identities P_i $(i = 0, 1, 2, ..., n)$; the delegation time period and the description of signing rights. Each proxy signer verifies the signature of the original signer for *w*.

Proxy Signature Generation: For obtaining the proxy signature on a message M_i, a proxy user $P_i \in L$ submits ID_i , private key of ID_i, message M_i, along with the warrant *w*, and *Params* as input; to this algorithm and outputs σ_i as a valid proxy signature.

Aggregation: On receiving different *n* proxy signatures $\{\sigma_i\}_{i=1, 2, ..., n}$, along with *n* identities, message pairs $\{ID_i, M_i\}_{i=1, 2, .., n}$, along with *w*, anyone among the proxy signers or a third party, can output σ as an aggregate proxy signature by running this algorithm.

Aggregate Proxy Signature Verification: This algorithm takes an aggregate proxy signature σ, the *n* identities, message pairs $\{ID_i, M_i\}_{i=1, 2, .., n}$, along with *w*, as input, verifies whether σ is valid or not. If true, it outputs '1', else output '0'.

3.2 Security Model of the Proposed IDAPS Scheme

In the following, we present the security model of our IDAPS scheme based on the security model in [19]. In this model, the following game played between the forger/adversary A and the challenger C. We divide the potential adversary A into the following three types.

Type 1 Adversary: In this type, the adversary A_1 is provided with the public keys of the original signer and all the proxy signers, and tries to forge the delegation for a chosen warrant or to forge the aggregate proxy signature for some chosen aggregate messages.

Type 2 Adversary: In this type, the adversary A_2 is provided with not only the public keys of the original signer and all the proxy signers, but also all the private keys of the proxy signers, and tries to forge the delegation by directly forging a valid signature for a chosen warrant.

Type 3 Adversary: In this type, the adversary A_3 is provided with not only the public keys of the original signer and all the proxy signers, but also the private key of the original signer, and tries to forge the aggregate proxy signature for some chosen aggregate messages.

It is to see that the aggregate proxy signature scheme can resist the attacks plotted from both the Type 2 and Type 3 adversaries, then it will be secure against the Type 1 adversary straight forwardly. The security model of our proposed scheme is defined in the following.

Defiinition 1: An aggregate proxy signature scheme is said to be secure against any Type 2 adversary if there is no probabilistic polynomial-time adversary A_2 can forge a valid signature σ_w on a chosen warrant *w* by playing the game with a challenger C. In addition, A_2 is said $(t, q_{H_1}, q_E, q_{H_2}, q_D, \varepsilon)-$to break a *N*-user IDAPS scheme if A_2 can run in time at most *t*; makes at most $q_{H_1} + q_{H_2}$ queries to the oracles

H_1, H_2; at most q_E queries to key extract query; at most q_D queries to delegation query; with $Adv_{IDAPS,\, A_2}$ is at least ε, in the game defined as follows.

Setup: C runs system setup phase to obtain *Params*.

H_1 – **Query:** C runs the H_1 oracle on a chosen identity ID_i and returns $H_1(ID_i)$.

H_2 – **Query:** C runs the H_1 oracle on a chosen identity ID_i, a chosen warrant *w*, and a random $U \in G$, and then returns $H_2(ID, w, U)$.

Key Extract Query: C runs the key extract phase on a chosen identity ID_i, and returns a private key corresponding to ID_i.

Delegation Query: C runs the delegation phase on a chosen warrant *w* and returns a signature σ_w of *w*.

Output: The adversary A_2 outputs $\{ID_0, w', \sigma'_w\}$ and wins the game if:

1. w' is not *w*; and
2. σ'_w is a valid signature of w'.

Defiinition 2: An aggregate proxy signature scheme is said to be secure against any Type 3 adversary if there is no probabilistic polynomial-time adversary A_3 can forge a valid aggregate proxy signature σ_{agg} on the chosen aggregate messages $\{M_i\}_{i=1,\, 2,\, \ldots,\, n}$ by playing the game with a challenger C. In addition, A_3 is said $(t, q_{H_1}, q_E, q_{H_2}, q_{H_3}, q_S, N, \varepsilon)-$to break a *N*-user IDAPS scheme if A_3 can run in time at most *t*; makes at most $q_{H_1} + q_{H_2} + q_{H_3}$ queries to the oracles H_1, H_2, H_3; at most q_E queries to key extract query; at most q_S queries to aggregate sign query; for obtaining at most *N* forged individual proxy signatures, with advantage $Adv_{IDAPS,\, A_3}$ is at least ε, in the game defined as follows.

The Setup, H_1, H_2, Key extract queries are same as defined above made by the adversary A_2.

H_3 – **Query:** C runs the H_3 oracle on a given warrant, chosen identity ID_i, a chosen message M_i, a chosen valid signing time T_i, a chosen signature $\sigma_0 = (U_0, V_0)$ of an original signer returns $H_3(ID_i, M_i, w, U_0, V_0)$.

Aggregate Signature Query: Given the identities, messages and signing times tuple $\{ID_i, M_i, T_i\}_{i=1,\, 2,\, \ldots,\, n}$ of *n* proxy signers, C runs proxy signature generation phase *n* times to obtain a proxy signature σ_i for M_i for (*i*=1, 2, …, *n*), and then runs the aggregation phase and returns a valid aggregate proxy signature σ_{agg} on the given $\{ID_i, M_i, T_i\}_{i=1,\, 2,\, \ldots,\, n}$.

Output: The adversary A_3 outputs $\{ID_i, M'_i, T_i, \sigma'_{agg}\}$ for $\{M'_i\}_{i=1,\, 2,\, \ldots,\, n}$ and wins the game if:

1. M'_i is not any of $M_1, M_2, \ldots, M_n$ and
2. σ'_{agg} is a valid aggregate proxy signature.

4. THE PROPOSED ID-BASED AGGREGATE PROXY SIGNATURE SCHEME

In this, we present the proposed IDAPS scheme and its detailed functionalities, as described in Section 3.1.

1. **System Setup:** For a given security parameter *l*, the KGC run this algorithm as follows:
   - Generate two cyclic groups $(G, +)$, $(G_T, \cdot)$ such that $|G| = |G_T| = q \geq 2^l$, q a prime.
   - Generate a generator $P \in G$ and an admissible bilinear map $\hat{e}: G \times G \to G_T$.
   - Picks an integer $s \in Z_q^*$ at random and computes $P_{pub} = sP$ as the system's overall public key. Also computes $g = \hat{e}(P_{pub}, P)$.
   - Picks hash functions $H_1: \{0, 1\}^* \to G$, $H_2: \{0, 1\}^* \times G_T \to Z_q^*$, and $H_3: \{0, 1\}^* \times G \times G_T \to Z_q^*$.

- Publishes the system's public parameters as $Params =< G,\ G_T,\ \hat{e},\ q,\ P,\ P_{pub},\ H_1,\ H_2,\ H_3,\ g >$ and keeps the system's master private key <s> with itself.

2. **Key Extraction:** This algorithm run by the KGC generates the public and private keys of a signer P_i with identity ID_i for i=1, 2, …, n. Upon receiving the identity ID_i KGC computes $Q_{ID_i} = H_1(ID_i) \in G$ as the public key of ID_i and $d_{ID_i} = sQ_{ID_i} \in G$ as the private key of ID_i and sends d_{ID_i} securely to ID_i.
3. **Warrant Delegation:** The original signer P_0 first prepares a warrant w to delegate his/her signing capability to the proxy signers $\{P_i\}_{i=1,\ 2,\ ...,\ n}$. The warrant w states the necessary proxy details, such as the identity information of the original signer ID_0, and of the n proxy signers $\{ID_i\}_{i=1,\ 2,\ ...,\ n}$, the form of information delegated, the period of delegation, i.e. the start-time T_S and the end-time T_E of the delegation. Now $w = \{ID_0,\ ID_1,\ ID_2,\ ...,\ ID_n,\ T_S,\ T_E\}$.
   The signer P_0 generates a signature $\sigma_0 = (U_0,\ V_0) \in G_T \times G$ for w by computing:
   $U_0 = g^{r_0} \in G_T$, for a random integer $r_0 \in Z_q^*$.
   $h_0 = H_2(ID_0,\ w,\ U_0) \in Z_q^*$,
   $V_0 = h_0 d_{ID_0} + r_0 P_{pub} \in G.$
   Finally, P_0 sends $\{ID_0,\ w,\ \sigma_0\}$ to each proxy signer P_i. Each proxy signer P_i can verify the validity of the signature σ_0 of w by checking the following equality.
   $\hat{e}(P,\ V_0) = \hat{e}(P_{pub},\ h_0 Q_{ID_0})U_0.$
   Proof of correctness:
   $$\begin{aligned}\hat{e}(P,\ V_0) &= \hat{e}(P,\ h_0 d_{ID_0} + r_0 P_{pub}) = \hat{e}(P,\ h_0 d_{ID_0})\hat{e}(P,\ r_0 P_{pub}) \\ &= \hat{e}(P,\ h_0 sQ_{ID_0})\hat{e}(P,\ r_0 sP) = \hat{e}(sP,\ h_0 Q_{ID_0})\hat{e}(sP,\ P)^{r_0} \\ &= \hat{e}(P_{pub},\ h_0 Q_{ID_0})\hat{e}(P_{pub},\ P)^{r_0} = \hat{e}(P_{pub},\ h_0 Q_{ID_0})U_0.\end{aligned}$$
4. **Proxy Signature Generation:** When the proxy signer P_i wants to sign the message M_i at the time T_i, for i=1, 2, …, n, under the warrant w, he/she verifies whether $T_S \le T_i \le T_E$ or not. If T_i is out of the valid period of the delegation, then abort this phase. Otherwise, P_i generates an individual proxy signature $\sigma_i = (U_i,\ V_i) \in G_T \times G$ for M_i by computing:
   $U_i = g^{r_i} \in G_T$
   $h_i = H_3(ID_i,\ M_i,\ w,\ V_0,\ U_0) \in Z_q^*$
   $V_i = h_i d_{ID_i} + r_i P_{pub} \in G.$
   Now, P_i sends the tuple $\{ID_i,\ M_i,\ T_i,\ \sigma_i\}$ to the aggregate phase.
5. **Aggregation:** Upon receiving $\{ID_i,\ M_i,\ T_i,\ \sigma_i\}$ sent by P_i, this algorithm first verifies whether $T_S \le T_i \le T_E$ or not. If T_i is out of the valid period of the delegation, then discard the proxy signature σ_i. Otherwise this algorithm computes $U = \prod_{i=1}^{n} U_i,\ V = \sum_{i=1}^{n} V_i$ and outputs the aggregate proxy signature $\sigma_{agg} = (U,\ V)$. Now this algorithm assures the validity of the individual proxy signature $\sigma_i = (U_i,\ V_i)$ of M_i by checking the following equation.
   $\hat{e}(P,\ V) = \hat{e}(P_{pub},\ \sum h_i Q_{ID_i})U.$
   Finally, this algorithm publishes $\{ID_0,\ w,\ \sigma_0,\ ID_i,\ M_i,\ T_i,\ \sigma_{agg}\}$ to the verifier (s).
6. **Aggregate Proxy Signature Verification:** Upon receiving $\{ID_0,\ w,\ \sigma_0,\ ID_i,\ M_i,\ T_i,\ \sigma_{agg}\}$, the verifier first verifies whether $T_S \le T_i \le T_E$ or not for all T_i's. if any T_i is out of the valid period of the

delegation, then decline the aggregate proxy signature σ_{agg}. Otherwise, the verifier ensures the validity of σ_{agg} for $\{M_i\}_{i=1, 2, ..., 3}$ by checking the following equation.

$$\hat{e}(P, V) = \hat{e}(P, \sum h_i d_{ID_i} + r_i P_{pub}) = \hat{e}(P, \sum h_i d_{ID_i})\hat{e}(P, \sum r_i P_{pub})$$
$$= \hat{e}(P, \sum h_i sQ_{ID_i})\hat{e}(P, \sum r_i sP) = \hat{e}(sP, \sum h_i Q_{ID_i})\prod \hat{e}(sP, P)^{r_i}$$
$$= \hat{e}(P_{pub}, \sum h_i Q_{ID_i})\prod \hat{e}(P_{pub}, P)^{r_i} = \hat{e}(P_{pub}, \sum h_i Q_{ID_i})U.$$

5. SECURITY ANALYSIS

In this, we prove the security of the proposed IDAPS scheme in the random oracle model, for a potential adversary of Type 2 and Type 3.

.

Theorem 1: Let A_2 is a probabilistic polynomial time forger who can forge the proposed IDAPS scheme with non negligible advantage. We show how to construct an algorithm B which can output the given CDH instance with non-negligible advantage in probabilistic polynomial time.

Proof: Let a forger A_2, breaks the proposed IDKIPS scheme. An algorithm say B is provided with $aP, bP \in G$ and its goal is to output $abP \in G$. B simulates an original signer to obtain a valid signature from A_2 and by doing so can solve the CDH problem.

Setup: B sets the system's overall public key as $P_{pub} = aP$ and starts by giving A_2 the Params. A_2 is also provided a randomly generated identity ID_1. From then onwards, A_2 can query the oracles H_1, H_2, H_3, Key Extract and delegation queries with B at any time.

H_1-**Queries:** B keeps a list L_1, which is empty initially, of tuples (ID_i, c_i, d_i, v_i) to respond to H_1-queries. Upon receiving a query on H_1 oracle for $ID \in \{0, 1\}^*$, made by A_2, B proceeds as follows:

1. If L_1 consists of the queried ID, then B responds with $H_1(ID) = v \in G$.
2. If not, B flips a coin $d \in \{0, 1\}$ generated at random, which outputs '0' with probability $1/(q_E + N)$.
3. Now, B picks a random integer $c \in Z_q^*$ and computes $v = c(bP) \in G$, for $d = 0$ and $v = cP \in G$, for $d = 1$.
4. B adds (ID, c, d, v) to the list L_1 and returns $H_1(ID) = v \in G$ to A_2.

H_2-**Queries:** B keeps a list L_2, which is empty initially, of tuples (ID, w, U, h), where w is a chosen warrant to respond to H_2 queries made by A_2. Upon receiving a query on tuple (ID_i, w, U_i), B proceeds as follows:

1. If L_2 is with the queried (ID_i, w, U_i), then B provides $H_2(ID_i, w, U_i) = h_i \in Z_q^*$.
2. If not, B picks an integer $h_i \in Z_q^*$ at random, inserts (ID_i, w, U_i, h_i) in L_2 and returns $H_2(ID_i, w, U_i) = h_i \in Z_q^*$ to A_2.

Key Extract Queries: Upon receiving the private key query on an identity ID_i by A_2, B retrieves the respective tuple (ID_i, c_i, d_i, v_i) from L_1 and does the following.

1. It outputs 'failure' and halts, for $d_i = 0$.
2. If not, computes and returns $d_{ID_i} = c_i P_{pub} = c_i(aP) = a(c_i P) \in G$ to A_2.

Delegation: Upon receiving A_2's query on a given warrant w_i for an original signer with the identity ID_i, B first confirms that $\{ID_i, w_i\}$ was not requested before. If $\{ID_i, w_i\}$ was requested before, then B returns failure and aborts, otherwise does the following.

1. Runs H_1 query on ID_i and get the corresponding instance of 4-tuple (ID_i, c_i, d_i, v_i) from L_1.

2. Computes $U_i = g^{k_i}$, where $k_i \in Z_q^*$ is chosen at random and $g = \hat{e}(P_{pub}, P)$.
3. Run H_2 query on (ID_i, w_i, U_i), and get the corresponding instance of 4-tuple (ID_i, w_i, U_i, h_i) from L_2.
4. If $d_i = 0$ holds, then return failure and abort, else compute $V_i = (h_i c_i + k_i)P_{pub}$ and return $\sigma_i = (U_i, V_i)$ as a signature for w_i.

Output: Eventually, A stops by conceding failure, as does B or returns a forgery $\sigma_i = (U_i, V_i)$ for the given warrant w_i under ID_i. Algorithm B obtains (ID_i, c_i, d_i, v_i) from L_1, declares failure if $d_i = 1$ and stops. If not, computes $Q_{ID_i} = c_i(bP)$, for $d_i = 0$. This forged signature σ_i must satisfy

$$\hat{e}(P, V_i) = \hat{e}(P_{pub}, h_i Q_{ID_i})U_i.$$

Now, B retrieves the respective tuple (ID_i, w_i, U_i, h_i) from L_2 and computes $V_i = (h_i c_i + k_i)P_{pub}$. for $i > 1$, we have

$$\begin{aligned}\hat{e}(P, V_i) &= \hat{e}(P, h_i Q_{ID_i} P_{pub})\hat{e}(P_{pub}, k_i P)\\ &= e(aP, h_i c_i(bP) + k_i P)\\ &= e(P, h_i c_i(abP) + k_i aP)\\ &= e(P, h_i c_i(abP) + k_i P_{pub})\end{aligned}$$

$$\Rightarrow V_i = h_i c_i abP + k_i P_{pub} \Rightarrow abP = h_i^{-1} c_i^{-1}(V_i - k_i P_{pub}).$$

This concludes the description of algorithm B.

Theorem 2: Let A_3 is a probabilistic polynomial time forger who can forge the proposed IDAPS scheme with non negligible advantage. We show how to construct an algorithm B which can output the given CDH instance with non-negligible advantage in probabilistic polynomial time.

Proof: Let a forger A_3, breaks the proposed IDKIPS scheme. An algorithm say B is provided with $aP, bP \in G$ and its goal is to output $abP \in G$. B simulates an original signer to obtain a valid signature from A_3 and by doing so can solve the CDH problem.

The setup phase, queries to the oracles H_1, H_2, key extraction, made by the forger A_3, is similar to that of the forger A_2, described in proof under Theorem 1. At any time, A_3 can make the queries to the oracles H_1, H_2, key extraction, H_3, and aggregate sign with B as follows:

H_3 − **Queries:** B keeps a list L_2, which is empty initially, of tuples $(ID_i, M, w, U_0, V_0, h_p)$ to respond to H_3 queries made by A_3. Upon receiving a query on tuple (ID_p, M_p, w, U_0, V_0), B proceeds as follows:

1. If L_2 is with the queried (ID_p, M_p, T, U_0, V_0), then B provides $H_3(ID_p, M_p, w, U_0, V_0) = h_p \in Z_q^*$.
2. If not, B picks a random integer $h_p \in Z_q^*$, inserts $(ID_p, M_p, w, U_0, V_0, h_p)$ in L_2 and returns $H_3(ID_p, M_p, w, U_0, V_0) = h_p \in Z_q^*$ to A_3.

Aggregate Sign Queries: By definition A_3 knows the private key of the original signer P_0 and has the ability to generate a forged valid signature $\sigma_0 = (U_0, V_0)$ for a chosen warrant w_0. When A_3 makes this query on given aggregate set of identities, messages, and sign time tuple, i.e. $(ID_i, M_i, T_i)_{i=1, 2, ..., n}$ of *n* proxy signers under the chosen warrant w_0, B first confirms (ID_i, M_i, T_i) has not been requested before. If (ID_i, M_i, T_i) was requested before, then B returns failure and aborts. Otherwise does the following on each ID_i and M_i for *i*=1, 2, …, *n*.

1. B queries the H_1 oracle and obtains (ID_i, c_i, d_i, v_i) from L_1, picks random integers $k_0, k_i \in Z_q^*$ and computes $U_0 = g^{k_0}$, $U_i = g^{k_i}$ where $g = \hat{e}(P_{pub}, P)$.
2. If L_2 contains (ID_0, w_0, U_0, h_0), then B picks $h_0' \in Z_q^*$ and tries again, i.e. B adds (ID_0, w_0, U_0, h_0), to L_2.

Now, B computes $V_0 = (h_0 c_0 + k_0)P_{pub}$ and returns $\sigma_0 = (U_0, V_0)$ to A_3 as the queried valid signature for warrant w_0. This can be seen from the following.

$$\begin{aligned}\hat{e}(P, V_0) &= \hat{e}(P, (h_0 c_0 + k_0)P_{pub}) \\ &= \hat{e}(P, h_0 c_0 P_{pub})\hat{e}(P, k_0 P_{pub}) \\ &= \hat{e}(aP, h_0 c_0 P)\hat{e}(aP, P)^{k_0} \\ &= \hat{e}(P_{pub}, h_0 Q_{ID_0})U_0.\end{aligned}$$

3. If L_3 contains $(ID_i, M_i, w_0, U_0, V_0, h_i)$, then B picks $h_i' \in Z_q^*$ and tries again, i.e. B adds $(ID_i, M_i, w_0, U_0, V_0, h_i')$, to L_3. Now, B computes $V_i = (h_i c_i + k_i)P_{pub}$ and returns $\sigma_i = (U_i, V_i)$ to A as the queried valid proxy signature of P_i with ID_i under warrant w_0. This can be seen from the equation: $\hat{e}(P, V_i) = \hat{e}(P_{pub}, h_i Q_{ID_i})U_i$.
4. If B does not abort any one of the queries and successfully outputs n forged individual proxy signatures (U_i, V_i) for i=1, 2, …, n, then B computes $U = \prod_{i=1}^{n} U_i$, $V = \sum_{i=1}^{n} V_i$ and returns (U, V).

Output: Eventually, A_3 stops by conceding failure, as does B or returns a aggregate forgery σ on the set of message, identity pairs $\{M_i, ID_i\}_{i=1, 2, ..., n}$, not querying a signature on M_1 under ID_1. Algorithm B obtains (ID_i, c_i, d_i, v_i) from L_1 and continues if $d_1 = 0$ and $d_i = 1$ for $2 \le i \le n$. If not, B declares failure and stops. We have $Q_{ID_1} = c_1(bP)$, for $d_1 = 0$ and $Q_{ID_i} = c_i P$, for $d_i = 1$, $i > 1$. This forged aggregate proxy signature σ must satisfy $\hat{e}(P, V) = \hat{e}(P_{pub}, \sum h_i Q_{ID_i})U$.

Now, B retrieves the n respective tuples $(ID_i, M_i, w_0, U_0, V_0, h_i)$, from L_3 and computes $V_i = (h_i c_i + k_i)P_{pub}$ for $i > 1$, we have

$$\hat{e}(P, V_i) = \hat{e}(P, (h_i c_i + k_i)P_{pub}) = e(P_{pub}, h_i Q_{ID_i})\hat{e}(P_{pub}, P)^{k_i} = e(P_{pub}, h_i Q_{ID_i})U_i.$$

Implies σ_i is valid.

Now, B considers $V_1 = V - \sum_{i=2}^{n} V_i$, and outputs

$$\begin{aligned}\hat{e}(P, V_1) &= \hat{e}(P, V - \sum_{i=2}^{n} V_i) = \hat{e}(P_{pub}, h_1 Q_{ID_1})U_1 \\ &= \hat{e}(aP, h_1 c_1(bP))\hat{e}(aP, P)^{k_1} = \hat{e}(P, h_1 c_1 abP + k_1 P_{pub}).\end{aligned}$$

$$\Rightarrow V_1 = h_1 c_1 abP + k_1 P_{pub} \Rightarrow h_1 c_1 abP = V_1 - k_1 P_{pub} \Rightarrow abP = h_1^{-1} c_1^{-1}(V_1 - k_1 P_{pub}).$$

This concludes the description of algorithm B.

6. EFFICIENCY ANALYSIS

To compare the computational and communication efficiency of the proposed IDAPS scheme, we consider the time-exhausting operations. According to [22, 23], $1T_p \approx 1200t_m$, $1T_m \approx 29t_m$, $1T_a \approx 0.12t_m$, where T_a denote the time for evaluating a point addition in G, T_m denote the time for evaluating a point scalar multiplication over G, T_p denotes the time to compute one pairing operation, and t_m denote the time to perform a modular multiplication in Z_q^*. Compared with the other operations, pairing evaluation is the most time expensive. Even much research [21] is taking place to speed up the pairing computation, it is still time consuming.

As shown in Table 1, the proposed IDAPS scheme requires a constant (two) number of pairing computations for aggregate verification, and is independent with the number of signers; and requires less pairing operations compared with the scheme [19]. Thus our scheme is computationally more efficient than the scheme [19].

Also from Table 1, the aggregate signature size of the proposed IDAS scheme is $2|G|$, which is independent of the number of signers. So, the proposed IDAS scheme is equally efficient, in communicational point of view, with the scheme [19]. But the security reduction in [19] is obtained using Forking lemma and so is not tightly related to the hard problem as pointed by the authors in [24, 25].

Table 1. Efficiency Table

Scheme	Aggregate Signature Size	Aggregation	Aggregate Verification
Lin et al. [19]	$2\|G\|$	$2(n-1)T_a \approx 0.24(n-1)t_m$	$3T_p + (n+2)T_m \approx (29n+3658)t_m$
Our IDAPS Scheme	$2\|G\|$	$(n-1)T_a \approx 0.12(n-1)t_m$	$2T_p + nT_m + (n-1)T_a \approx (29.12n+2399.88)t_m$

7. CONCLUSION

This paper proposed a new and efficient IDAPS scheme using pairings over elliptic curves. This scheme achieves constant aggregate proxy signature size and requires a constant (two) number of pairing computations in aggregate verification. In the random oracle paradigm, the proposed scheme is unforgeable, proven secure under CDH assumption without using Forking lemma. From the efficiency analysis of our scheme, we conclude that the proposed scheme is more efficient than the related schemes of this kind in terms of computational overhead and communication bandwidth.

REFERENCES

[1] M. Mambo, et al., “Proxy Signatures: Delegation of the Power to Sign Messages,” *IEICE Transactions on Fundamentals of Electronic Communications and Computer Science*, Vol. E79-A, No. 9, pp. 1338-1354, 1996.

[2] D. Boneh, et al., “Aggregate and Verifiably encrypted Signatures from Bilinear Maps,” *Eurocrypt 2003*, LNCS 2656, Springer Verlag, pp. 416-432, 2003.

[3] A. Shamir, “Identity-based Cryptosystems and Signature Schemes,” *Crypto 1984*, LNCS 196, Springer Verlag, pp. 47-53, 1984.

[4] D. Boneh, and M. Franklin, “Identity-based Encryption from the Weil Pairing,” *SIAM Journal of Computing*, Vol. 32(3), pp. 586-615, 2003.

[5] P.V.S.S.N. Gopal, et al., “New Identity Based Signature Scheme using Bilinear Pairings over Elliptic Curves,” *3rd IEEE International Advanced Computing Conference 2013*, pp. 362-367, 2012.

[6] F. Hess, “Efficient Identity Based Signature Schemes Based on Pairings,” LNCS 2595, Springer-Verlag, pp. 310-324, 2002.

[7] K.G. Paterson, “ID-Based Signatures From Pairings on Elliptic Curves,” *IEEE Electronic Letters*, Vol. 38, No. 18, pp. 1025-1026, 2002.

[8] J.C. Cha, et al., “An Identity-Based Signature Scheme from Gap Diffie-Hellman Groups,” *PKC* 2003, Springer-Verlag, LNCS 2567, pp. 18-30, 2003.

[9] J.H. Cheon, et al., “A New ID-Based Signature with Batch Verification,” Available: http://eprint.iacr.org/2004/131.

[10] J. Xu, et al., "ID-based Aggregate Signatures from Bilinear Pairings," *4th Int. Conference on Cryptology and Network Security*, LNCS 3810, Springer-Verlag, pp. 110-119, 2005.
[11] C. Gentry, et al., "Identity-based Aggregate Signatures," *PKC 2006*, LNCS 3958, Springer-Verlag, pp. 257-273, 2006.
[12] F. Zhang, et al., "Efficient ID-based Blind Signature and Proxy Signature from Bilinear Pairings," *8th Australasia Conference on Information Security and Privacy* (ACISP 2003), Vol. 2727, pp. 312-323, 2003.
[13] K.A. Shim, An ID-based Aggregate Signature Scheme with Constant Pairing Computations, *The Journal of Systems and Software*, Vol. 83, pp. 1873-1880, 2010.
[14] Y. Yu, X. Zheng, and H. Sun, "An Identity Based Aggregate Signature from Pairings," *Journal of Networks*, vol. 6, No. 4, pp. 631-637, 2011.
[15] J. Xu, et al., "ID-based Proxy Signature using Bilinear Pairings," *International Symposium on Parallel and Distributed Processing and Applications*, LNCS 3759, pp. 359-367, 2005.
[16] W. Wu, et al., "Identity-based Proxy Signature from Pairing," *4th International Conference on Automatic and Trusted Computing*, LNCS 4610, pp. 22-31, 2007.
[17] F. Cao, and Z. Cao, "A Secure Identity-based Multi-Proxy Signature Scheme," *Computers and Electrical Engineering*, vol. 35, pp. 86-95, 2009.
[18] R.A. Sahu, and S. Padhye, "Provable Secure Identity-based Multi-Proxy Signature Scheme," *International Journal of Communication Systems*, pp. , 2013.
[19] Y.C. Lin, T.C. Wu, and J.L. Tsai, "ID-Based Aggregate Proxy Signature Scheme Realizing Warrant-Based Delegation," *Journal of Information Science and Engineering*, vol. 29, pp. 441-457, 2013.
[20] D. Pointcheval, and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, 13(3), pp. 361-396, 2000.
[21] P. Barreto, et al., "Efficient algorithms for pairing based cryptosystems," *Crypto 2002*, LNCS 2442, Springer-Verlag, pp.354-368, 2002.
[22] N. Koblitz, A. Menezes, and S. Vanstone, "The State of Elliptic Curve Cryptography," *Designs, Codes and Cryptography*, 19, 2-3, pp. 173-193, 2000.
[23] A. Menezes, P. Van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography," CRC Press, LLC, Boca Raton, Fla, USA, 1997.
[24] E.J. Goh, et al., "A signature scheme as secure as the Diffie-Hellman problem," LNCS 2656, Springer-Verlag, pp. 401-415, 2003.
[25] J. Katz, et al., "Efficiency improvements for signature schemes with tight security reductions," *10th ACM CCS Conference*, pp. 155-164, 2003.

BIBLIOGRAPHY OF AUTHORS

P.V.S.S.N. Gopal received the M.Sc degree in Applied Mathematics and M.Phil in (Commutative Algebra) Mathematics from Pondicherry University, Pondicherry, India in 2001 and 2008 respectively. He is presently pursuing Ph.D in Andhra University, Visakhapatnam, India. His area of interests includes Elliptic Curve Cryptography, Abstract Algebra, Linear Algebra & Number theory.

P. Vasudeva Reddy received M.Sc (Mathematics), Ph.D (Cryptography) from S.V. University, Tirupati, M. Tech (CST-Networks) from Andhra University, India. He is currently working as an Associate professor in the department of Engg. Mathematics, College of Engineering, Andhra University, Visakhapatnam, India. His field of interest includes Algebra & Number theory Applications, secret sharing, and Cryptography. He has several publications in national and international reputed journals. He is a life member of Cryptology Research Society of India (CRSI).

T. Gowri received B.Tech from Nagarjuna University, and M. Tech from Jawaharlal Nehru Technological University. She is currently working as an Associate professor in the department of Electronics and Communication Engineering, GIET, GITAM University, Visakhapatnam, A.P, India. Her research interests include Digital Information Systems and Computer Electronics, Digital Image Processing and Information Security.