Key-Insulated Aggregate Proxy Signature

P. V. S. S. N. Gopal, T. Gowri, and P. Vasudeva Reddy

1 Introduction

Ever since, Public Key Cryptosystem (PKC) devised by Diffie and Hellman [1] in 1976, the cryptographic research took a rapid progress. Shamir [2] devised the notion of Identity-based PKC (IPKC), in 1984. In such cryptosystem, signer's public key comprises of binary sequence linked to their identity, like name, mobile number etc. Accordingly, the public key is verified explicitly without accompanying the matching public key certificate. Further, private keys are issued by the trusted party, termed the Key Control Centre (KCC). By the invention of IPKC, many encryption and signature schemes with bilinear pairings of elliptic curves were constructed [3, 4].

Most of the schemes were constructed under opinion that the private keys remain perfectly secure. The whole system's security will no longer be confidential, if suppose the KCC is compromised. To overcome such situation, Dodis et al. [5], devised a cryptosystem via key-insulated mechanism, in 2002.

The basic structure of the system [5] is split life time of master private key as distinct time periods, in which the long term private keys not used for signing directly called helper keys are maintained by a device that is physically-secure, the helper. To perform cryptographic operations, the signers store their interim private keys in a powerful but computationally limited device. Further, this mechanism revives

P. V. S. S. N. Gopal (✉)
Department of BS & H (Mathematics), Kallam Haranadhareddy Institute of Technology (A), Guntur, Andhra Pradesh 522019, India
e-mail: gopalcrypto786@gmail.com

T. Gowri
Department of EECE, GITAM Deemed to Be University, Visakhapatnam, Andhra Pradesh 530045, India

P. V. Reddy
Department of Engineering Mathematics, AU College of Engineering, Andhra University, Visakhapatnam, Andhra Pradesh 530003, India

B. K. Roy et al. (eds.), *Cryptology and Network Security with Machine Learning*, Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-99-2229-1_4

the momentary private key on distinct time periods through an interaction involving signer and helper; keeping public key unaffected all over. Thus, a compromise of some periods leaves the other unharmed. Hence, this mechanism effectively minimizes the harm caused by revelation of private key until it changes.

Based on scheme [5], the first signature scheme in Identity-based framework using key-insulated mechanism was constructed by Zhou et al. [6], in 2006. Later, many signature schemes and their extensions were constructed [7–9].

Boneh et al. [10], devised an aggregate signature, in 2003, which is single compressed signature attained on combining different n (signatures; signers; messages). Such signature is verified by anyone; convince themselves that the n signer's undeniably signed the n original messages.

Mambo et al. [11] devised a proxy signature in PKI based setting, in 1996. Later, Zhang et al. [12] constructed the first proxy signature scheme in 2003, in ID-based framework. In such a scheme, proxy signer signs on message in support of original signer, attained on receiving a warrant consisting of implicit description of signing rights issued to the former by the latter. Tiwari et al. [13] carried out an analysis on generalization of the proxy signature in 2013.

Wan et al. [14], in 2009, presented a Proxy Signature scheme using Key-insulted mechanism in Identity-based framework (IKPS), that needs 4 pairing computations in proxy signature verification phase proven secure in random oracle paradigm without use of Forking lemma [15].

Lin et al. [16], in 2013, presented an Aggregate Proxy Signature scheme in Identity-based framework (IAPS) on realizing warrant-based delegation. This scheme needs 3 pairing computations in the aggregate signature verification phase and uses Forking lemma [15] in its security reduction.

To handle the issues of key disclosure in proxy signature and maintaining the merits of aggregate signatures, in this article, we construct the first efficient Key-insulated Aggregate Proxy Signature scheme in Identity-based framework (IKAPS) that uses bilinear pairings of elliptic curves. The constructed scheme involves only 3 (constant) pairing calculations in its key-insulated aggregate proxy signature verification phase. Further, we demonstrate that the constructed scheme's security is tightly secure to the hardness of Computational Diffie-Hellman problem [17, 18] in random oracle paradigm without the use of Forking lemma [15].

The rest of paper is categorized as follows: devoted Sect. 2, to some preliminaries including computational hard problems. The constructed IKAPS scheme along with schematic diagram is exhibited under Sect. 3. The constructed scheme's security and its proof of correctness are exhibited in Sect. 4. Efficiency analysis of the constructed scheme is depicted in Sect. 5 and conclusion exhibited finally under Sect. 6.

2 Preliminaries

We summarize the symbolizations and their depiction used in the work; some essential notions; necessary hard problems under this section.

Table 1 Various symbolizations and their depiction used in the constructed scheme

Symbolizations	Depiction
$\mathcal{G}_a$	Additive cyclic group
$\mathcal{G}_m$	Multiplicative cyclic group
$\in_R$	Picked at random from the respective set
$\|\mathcal{G}\|$	Order of group
ID_i	The signer S_i's identity
d_{ID_i}	The ID_i's private key
$PSIK_{ID_i,0}$	Proxy signer's initial private key
$HPK_{ID_i,\, t}$	Proxy helper's private key in t a time period
$PSUK_{ID_i,\, t}$	Proxy signer's update signing key in time period t
$\{S_i\}_{i=1,\,2,\,\ldots,\,n}$	An aggregate collection of proxy signers
$\{M_i\}_{i=1,\,2,\,\ldots,\,n}$	An aggregate collection of messages
σ_i	A key-insulated proxy signature on the message M_i by S_i
$\{\sigma_i\}_{i=1,\,2,\,\ldots,\,n}$	An aggregate collection of key-insulated proxy signatures
σ	A key-insulated aggregate proxy signature

2.1 Symbolizations and Their Depiction Used in the Constructed Scheme

The symbolizations and their depiction used in the constructed scheme are presented in the following Table 1.

2.2 Bilinear Map

Let $(\mathcal{G}_a,\, +)$, $(\mathcal{G}_m,\, \cdot)$ be as mentioned in 2.1, of equal prime order q, and P (say) generates $\mathcal{G}_a$. A function $e : \mathcal{G}_a \times \mathcal{G}_a \to \mathcal{G}_m$ is called bilinear map if the below laws are satisfied:

I. **Bilinear:** $\forall\, U,\ V \in \mathcal{G}_a, \forall\, x,\ y \in_R Z_q^*, e(xU,\ yV) = e(U,\ V)^{xy}$.
II. **Non-Degeneracy:** $\exists\, U \in \mathcal{G}_a, \ni\ e(U,\ U) \neq 1$.
III. **Calculable:** $\forall\, U,\ V \in \mathcal{G}_a, e(U,\ V)$ is calculated by effective algorithm.

On formulating appropriate modifications in Weil/Tate pairing, one works on such using elliptic curves of finite fields.

2.3 Complexity Assumptions

We now exhibit some compulsory hard problems which are used in the constructed scheme's security reduction, in the following.

- **Computational Diffie-Hellman (CDH) Problem:** $\forall c, d \in Z_q^*$, given $P, cP, dP \in \mathcal{G}_a$ evaluate $cdP \in \mathcal{G}_a$. For $\mathcal{A}$, a adversary in polynomial-time is of advantage (Adv) described as t, the run time in opposition to the CDH problem in $\mathcal{G}_a$, i.e., $Adv_{CDH}(t) = \Pr[\mathcal{A}(P, cP, dP) = cdP/P, cP, dP \in \mathcal{G}_a]$.
- **Computational Diffie-Hellman (CDH) Assumption:** the $(t, \varepsilon)-$ CDH assumption believed to hold in the group $\mathcal{G}_a$ if no $\mathcal{A}$ with Adv at least ε in t-time can break the CDH problem.

3 The Constructed IKAPS Scheme and Its Schematic Diagram

This section refers to the constructed IKAPS scheme, which involves eight algorithms as portrayed below.

1. **Setup:** For $l \in_R Z^+$ security parameter, the KCC run the setup algorithm as portrayed below:
   - Picks two cyclic groups $\mathcal{G}_a, \mathcal{G}_m$ under the binary operations addition, multiplication respectively, of same prime order say $q \geq 2^l$.
   - Picks P a generator of $\mathcal{G}_a$ and $e : \mathcal{G}_a \times \mathcal{G}_a \rightarrow \mathcal{G}_m$ a bilinear map.
   - Picks $s, hpk \in_R Z_q^*$, calculates $P_{pub} = sP,\ P_{hlp} = hpkP$ as appropriate overall system's, helper's public keys, $\text{g} = e(P_{pub}, P)$.
   - Picks the hash functions $\mathcal{H}_1, \mathcal{H}_2 : \{0, 1\}^* \rightarrow \mathcal{G}_a$, $\mathcal{H}_3 : \{0, 1\}^* \times \mathcal{G}_m \rightarrow Z_q^*$, $\mathcal{H}_4 : \{0, 1\}^* \times \mathcal{G}_a \times \mathcal{G}_m \rightarrow Z_q^*$.
   - Publishes the system's parameters which are made public as $\mathcal{PP} =< l, \mathcal{G}_a, \mathcal{G}_m, q, P, e, P_{pub}, P_{hlp}, \text{g}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4 >$, holds $<s>$, $<hpk>$ with itself securely.
2. **Key Ext:** The KCC run this algorithm to produce public and private keys of a signer S_i with identity ID_i for $i = 0, 1, 2, ..., n$. On attaining ID_i of S_i, it calculates $Q_{ID_i} = H_1(ID_i)$, $d_{ID_i} = sQ_{ID_i}$ as appropriate public, private keys of S_i, sends d_{ID_i} to S_i securely.
3. **Initial Proxy Key Gen:** The KCC and the original signer carry out this algorithm. At first, S_0 the original signer prepares a warrant ω with all the necessary information about the allocation rights to the proxy signers $\{S_i\}_{i=1,2,...,n}$. The signer S_0 creates a signature $\sigma_0 = (\mathcal{U}_0, \mathcal{V}_0)$ for ω on calculating $\mathcal{U}_0 = \text{g}^{r_0}$ where $r_0 \in_R Z_q^*$, $h_0 = \mathcal{H}_3(ID_0, M, \omega, U_0)$ and $\mathcal{V}_0 = h_0 d_{ID_0} + r_0 P_{pub}$. Finally, S_0 sends $\{ID_0, \omega, \sigma_0\}$ to each proxy signer S_i. Now, S_i can verify the authenticity of σ_0 as below:

$$e(P,\ \mathcal{V}_0) = e(P,\ h_0 d_{ID_0} + r_0 P_{pub}) = e(P_{pub},\ h_0 \mathcal{H}_1(ID_0))\mathcal{U}_0.$$

Now, KCC calculates $PSIK_{ID_i,0} = hd_{ID_i} + hpk\mathcal{H}_2(ID_i, 0)$ where $h = \mathcal{H}_4(ID_i,\ \mathcal{U}_0,\ \mathcal{V}_0,\ \omega)$, transmits $PSIK_{ID_i,0}$, <*hpk*> appropriate to proxy signer as their initial proxy signing key and helper as their helper private key securely. Here, '0' of $PSIK_{ID_i,0}$, denote the initial time period.

4. **Proxy Key Upd:**
   - **Helper Key Upd:** At time period t, helper of the proxy signer S_i, calculates a helper key $HPK_{ID_i,\,t} = hpk[\mathcal{H}_2(ID_i,\ t) - \mathcal{H}_2(ID_i,\ t-1)]$, forwards it to S_i.
   - **Proxy Signer Key Upd:** Now, S_i updates their private key $PSUK_{ID_i,\,t} = HPK_{ID_i,\,t} + PSIK_{ID_i,\,t-1}$. Finally, the proxy signer wipe away the values $HPK_{ID_i,\,t}$ and $PSIK_{ID_i,\,t-1}$.
5. **Key-insulated Proxy Sign Gen:** On acquiring message $\mathcal{M} \in \{0,\ 1\}^*$, in time period t, proxy signer S_i works as below:
   - Picks an integer $r_i \in_R Z_q^*$, and calculates

$$\mathcal{U}_i = g^{r_i}, \quad h = \mathcal{H}_4(ID_i,\ \mathcal{U}_0,\ \mathcal{V}_0,\ \omega),\ h_i = \mathcal{H}_4(ID_i,\ \mathcal{M},\ \omega,\ \mathcal{U}_0,\ \mathcal{V}_0,\ t),$$
$$\mathcal{V}_i = h_i PSUK_{ID_i,t} + r_i P_{pub}.$$

   - Outputs $\sigma_i = (\mathcal{U}_i,\ \mathcal{V}_i)$ the key-insulated proxy signature (IKPS) on $\mathcal{M}$, signed by S_i in t.
6. **Key-insulated Proxy Sign Ver:** Any signer run this algorithm that takes message, identity pairs $(\mathcal{M}_i,\ ID_i)$, key-insulated proxy signature $(\sigma_i,\ t)$ as input. The verification is done as follows:
   - Calculates $h = \mathcal{H}_4(ID_i, \mathcal{U}_0,\ \mathcal{V}_0,\ \omega),\ h_i = \mathcal{H}_4(ID_i,\ \mathcal{M},\ \omega, \mathcal{U}_0,\ \mathcal{V}_0,\ t)$.
   - Verify $e(P,\ \mathcal{V}_i) = e(P_{hlp},\ h_i\mathcal{H}_2(ID_i, t))e(P_{pub},\ hh_i\mathcal{H}_1(ID_i))\mathcal{U}_i$ valid or not. It outputs '1', for σ_i valid, else '0'.
7. **Key-insulated Agg Proxy Sign Gen:** Each proxy signer $\{S_i\}_{i=1,2,\ldots,n}$ presents their key-insulated proxy signature $(\sigma_i,\ t)$ in t. Now, any authorized signer calculates $\mathcal{U} = \Pi_{i=1}^{n}\mathcal{U}_i,\ \mathcal{V} = \Sigma_{i=1}^{n}\mathcal{V}_i$ and outputs $\sigma = (\mathcal{U},\ \mathcal{V})$ as the IKAPS.
8. **Key-insulated Agg Proxy Sign Ver:** Any verifier verifies IKAPS $(\sigma,\ t)$ for 't' as follows.
   - Calculates $h = \mathcal{H}_4(ID_i, \mathcal{U}_0,\ \mathcal{V}_0,\ \omega),\ h_i = \mathcal{H}_4(ID_i,\ \mathcal{M},\ \omega, \mathcal{U}_0,\ \mathcal{V}_0,\ t)$
   - Verify $e(P,\ \mathcal{V}) = e(P_{hlp},\ h_i\mathcal{H}_2(ID_i, t))e(P_{pub},\ hh_i\mathcal{H}_1(ID_i))\mathcal{U}$ for validity. It outputs '1', for σ valid, else '0'.

Now, we present the schematic diagram of IKAPS scheme (Fig. 1).

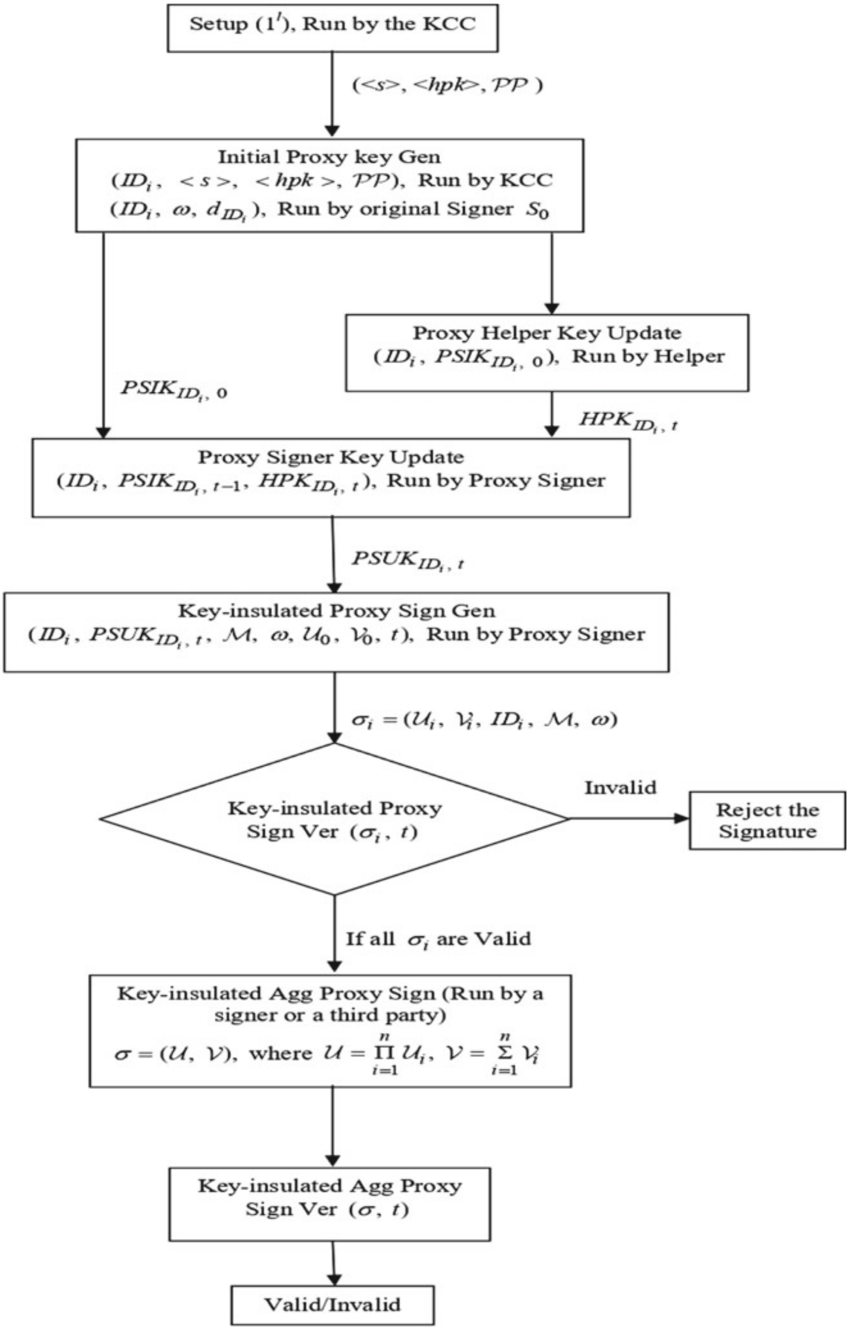

Fig. 1 Schematic diagram of the Constructed IKAPS Scheme

4 Security Analysis

This section briefs proof of correctness as well the security reduction, of constructed IKAPS scheme.

4.1 *Proof of Correctness*

For IKPS:

$$e(P,\ \mathcal{V}_i) = e(P,\ h_i PSUK_{ID_i,t} + r_i P_{pub}) \\ = e(P_{hlp},\ h_i \mathcal{H}_2(ID_i, t)) e(P_{pub},\ hh_i \mathcal{H}_1(ID_i)) \mathcal{U}_i.$$

For IKAPS:

$$e(P,\ \mathcal{V}) = e(P,\ \Sigma(h_i PSUK_{ID_i,t} + r_i P_{pub})) \\ = e(P_{hlp},\ \Sigma h_i \mathcal{H}_2(ID_i, t)) e(P_{pub},\ \Sigma hh_i \mathcal{H}_1(ID_i)) \mathcal{U}.$$

4.1.1 Security Reduction

Theorem: Assume $\mathcal{A}$ a forger, in polynomial time can forge the constructed IKAPS scheme with non-insignificant *Adv*. Next, there is some $\mathcal{B}$ an algorithm, which can output given CDH instance with the same *Adv* and time.

Proof: Let $\mathcal{A}$ cracks the constructed IKAPS scheme. An algorithm say $\mathcal{B}$ is provided with $xP,\ yP \in \mathcal{G}_a$ and its objective is to output $xyP \in \mathcal{G}_a$. For this, $\mathcal{B}$ replicates proxy signer to attain valid proxy signature from $\mathcal{A}$, to solve the CDH problem.

Setup: $\mathcal{B}$ puts $P_{pub} = xP$, forwards the $\mathcal{PP}$ to $\mathcal{A}$.

Queries: $\mathcal{A}$ queries $\{\mathcal{H}_i\}_{i=1,2,3,4}$ hash functions, proxy key gen and proxy sign ver at their convenience. We presume that before making any initial proxy private key, proxy signing queries on *ID*; $\mathcal{H}_1$ query was made on it earlier. For responding to such, $\mathcal{B}$ evolves as below.

– $\mathcal{H}_1-$ **Queries:** $\mathcal{B}$ possesses a list $\mathcal{L}_1$, empty initially, $(ID,\ c,\ d,\ v)$ of tuples to evolve with the queried hash $\mathcal{H}_1$ function. On getting a query on $\mathcal{H}_1$ for $ID \in \{0,\ 1\}^*$, by $\mathcal{A}$, $\mathcal{B}$ evolves as below.

1. If $\mathcal{L}_1$ comprises queried *ID*, $\mathcal{B}$ evolves with $\mathcal{H}_1(ID) = v$.
2. Else, $\mathcal{B}$ flips arbitrary coin $d \in \{0,\ 1\}$ with $\Pr[d = 0] = \frac{1}{q_{KE}+q_s+N}$.
3. Now, $\mathcal{B}$ picks $c \in_R Z_q^*$, for $d = 0$ calculates $v = c(yP)$ and $v = cP$ for $d = 1$.
4. Inserts $(ID,\ c,\ d,\ v)$ to $\mathcal{L}_1$, forwards $\mathcal{H}_1(ID) = v$ to $\mathcal{A}$.

- $\mathcal{H}_2-$ **Queries:** $\mathcal{B}$ possesses a list $\mathcal{L}_2$, empty initially, of tuples $(ID_f,\ t,\ k,\ kP)$, to evolve with the queried hash $\mathcal{H}_2$ function by $\mathcal{A}$. On getting a query on $(ID_f,\ t)$ by $\mathcal{A}$,$\mathcal{B}$ evolves as below.

  1. If $\mathcal{L}_2$ comprises the queried tuple, then $\mathcal{B}$ evolves with $\mathcal{H}_2(ID_f,\ t)$.
  2. Else, $\mathcal{B}$ picks $k\ \in_R\ Z_q^*$, calculates $\mathcal{H}_2(ID_f,\ t) = kP$, inserts $(ID_f,\ t,\ k,\ kP)$ to $\mathcal{L}_2$, forwards kP to $\mathcal{A}$.

- $\mathcal{H}_3-$ **Queries:** $\mathcal{B}$ possesses a list $\mathcal{L}_3$ of tuples $(ID_e,\ \omega,\ \mathcal{U}_e,\ h_3)$. On getting a query by $\mathcal{A}$ on H_3, $\mathcal{B}$ picks $h_3\ \in_R\ Z_q^*$, calculates $\mathcal{H}_3(ID_e,\ \omega,\ \mathcal{U}_e) = h_3$, inserts to $\mathcal{L}_3$, forwards h_3 to $\mathcal{A}$.
- $\mathcal{H}_4-$ **Queries:** $\mathcal{B}$ possesses a list $\mathcal{L}_4$ of tuples $(ID_f, \mathcal{U}_e,\ \mathcal{V}_e,\omega,\ h_4)$, empty initially, to evolve with the queried hash $\mathcal{H}_4$ function. On getting a query on $(ID_f,\ \mathcal{U}_e,\ \mathcal{V}_e,\omega)$ by $\mathcal{A}$,$\mathcal{B}$ evolves as below.

  1. If L_4 comprises the queried tuple, then $\mathcal{B}$ evolves with $\mathcal{H}_4(ID_f,\ \mathcal{U}_e,\ \mathcal{V}_e,\omega) = h_4$.
  2. Else, $\mathcal{B}$ picks $h_4\ \in_R\ Z_q^*$, calculates $\mathcal{H}_4(ID_f,\ \mathcal{U}_e,\ \mathcal{V}_e,\omega)\ =\ h_4$, inserts to $\mathcal{L}_4$ forwards to $\mathcal{A}$.

  Also, $\mathcal{B}$ possesses a list $\mathcal{L}_5$ of tuples $(ID_f,\ \mathcal{M},\ \omega,\ \mathcal{U}_e,\ \mathcal{V}_e,t,\ v\prime)$ empty initially, to evolve with the queried hash $\mathcal{H}_4$ function. On getting a query on $(ID_f,\ \mathcal{M},\ \omega,\ \mathcal{U}_e,\ \mathcal{V}_e,t)$ by $\mathcal{A}$,$\mathcal{B}$ evolves as below.

  1. If $\mathcal{L}_5$ comprises the queried tuple, then $\mathcal{B}$ evolves with $\mathcal{H}_4(ID_f,\ \mathcal{M},\ \omega,\ \mathcal{U}_e,\ \mathcal{V}_e,t) = v\prime$.
  2. Else, $\mathcal{B}$ picks $v\prime \in_R Z_q^*$, calculates $\mathcal{H}_4(ID_f,\ \mathcal{M},\ \omega,\ \mathcal{U}_e,\ \mathcal{V}_e,t) = v\prime$, inserts to $\mathcal{L}_5$, forwards $v\prime$ to $\mathcal{A}$.

- **Initial Proxy Key Queries:** $\mathcal{B}$ possesses a list $\mathcal{L}_6$, empty initially. On getting a query to $\{(ID_e,\ ID_f),\ \omega\}$ by $\mathcal{A}$,$\mathcal{B}$ evolves as below.

  1. $\mathcal{B}$ retrieve the tuples $(ID_e,\ c_e,\ d_e,\ v_e),\ (ID_f,\ c_f,\ d_f,\ v_f)$ from the list $\mathcal{L}_1$. If $d_e = 0$ or $d_f\ = 0$,$\mathcal{B}$ halts and outputs failure.
  2. Else, it infers that $\mathcal{H}_1(ID_e) = c_eP$ and $\mathcal{H}_1(ID_f) = c_fP$ as determined earlier.
  3. Now, $\mathcal{B}$ retrieve the tuples $(ID_f,\ c_f,\ d_f,\ v_f),(ID_f,\ t),(ID_f,\ \mathcal{U}_e,\ \mathcal{V}_e,\omega)$ from $\mathcal{L}_1,\ \mathcal{L}_2,\ \mathcal{L}_4$ respectively, calculates $d_{ID_f,0} = c_0P_{pub} + k_0P_{hlp}$, forwards $d_{ID_f,0}$ to $\mathcal{A}$.

     Now, $\mathcal{B}$ inserts $(ID_f,\ d_{ID_f,0})$ to $\mathcal{L}_6$.

- **Helper Key Update Query:** $\mathcal{B}$ possesses a list $\mathcal{L}_7$, empty initially. On getting a helper key query on ID_f by $\mathcal{A}$, in t, $\mathcal{B}$ retrieve $(ID_f,\ t,\ k,\ kP)$ from $\mathcal{L}_2$, calculates $HPK_{ID_f,t} = hpk(k_tP - k_{t-1}P)$, forwards $HPK_{ID_f,t}$ to $\mathcal{A}$.

  Now, $\mathcal{B}$ inserts $(ID_f,\ HPK_{ID_f,t})$ to $\mathcal{L}_7$.

– **Proxy Key Update Query:** $\mathcal{B}$ possesses a list $\mathcal{L}_8$, empty initially. On getting a update key query of a proxy signer ID_f by $\mathcal{A}$, in a time period t, $\mathcal{B}$ evolves as below.

1. $\mathcal{B}$ retrieves $(ID_f,\ d_{ID_f,0}), (ID_f,\ HPK_{ID_f,t})$ from $\mathcal{L}_6,\ \mathcal{L}_7$ respectively.
2. Calculates $d_{ID_f,t} = c_f P_{pub} + k_f P_{hlp}$.

– **Proxy Sign Queries:** On getting query to $((ID_e,\ ID_f),\ \mathcal{M},\ \omega,\ t)$, i.e., the proxy signature on $\mathcal{M}$ with warrant ω for ID_f by $\mathcal{A}$, in t, $\mathcal{B}$ evolves as below.

1. Picks $n_e, n_f \in_R Z_q^*$, calculates $\mathcal{U}_e = g^{n_e}, \mathcal{U}_f = g^{n_f}$ inserts $(ID_e,\ \omega,\ \mathcal{U}_e,\ h_3),\ (ID_f,\ \mathcal{U}_e,\ \mathcal{V}_e, \omega,\ h_4)$ to $\mathcal{L}_3,\ \mathcal{L}_4$ respectively.
2. Examines $\mathcal{L}_5$ for $(ID_f,\ \mathcal{M},\ \omega,\ \mathcal{U}_e,\ \mathcal{V}_e, t,\ v\prime)$ and retrieve the value determined earlier.
3. Examines $\mathcal{L}_8$ for $(ID_f,\ d_{ID_f,t})$ and retrieve the value determined earlier.
4. Fixes $\mathcal{V} = v\prime(h_4 c_f P_{pub} + k_f P_{hlp}) + n_f P_{pub}$.
5. Forwards to $\mathcal{A}$, the queried proxy signature $\sigma = (\mathcal{U},\ \mathcal{V})$. Answers to the proxy sign queries are all valid and also the output σ as observed below.

$$\begin{aligned} e(P,\ \mathcal{V}_f) &= e(P,\ v\prime(h_4 c_f P_{pub} + k_f P_{hlp}) + n_f P_{pub}) \\ &= e(P_{pub},\ v\prime h_4 \mathcal{H}_1(ID_f))\hat{e}(P_{hlp},\ v\prime \mathcal{H}_2(ID_f,\ t))\mathcal{U}_f. \end{aligned}$$

– **Output:** Ultimately, $\mathcal{A}$ on admitting failure halts, as $\mathcal{B}$ does, or returns a forged aggregate proxy signature $\sigma*$, on $\mathcal{M}*$, in $t*$. $\mathcal{B}$ retrieves $(ID_e,\ c_e,\ d_e,\ v_e),\ (ID_f,\ c_f,\ d_f,\ v_f)$ from $\mathcal{L}_1$. If $d_e^* = 1$ or then $\mathcal{B}$ output fails. Else, retrieves $(ID_e^*,\ \omega,\ \mathcal{U}_e^*,\ h_3^*), (ID_f^*,\ \mathcal{U}_e^*,\ \mathcal{V}_e^*, \omega,\ h_4^*), (ID_f^*,\ \mathcal{M}*,\ \omega,\ \mathcal{U}_e^*,\ \mathcal{V}_e^*, t*,\ v\prime)$ from $\mathcal{L}_3,\ \mathcal{L}_4,\ \mathcal{L}_5$ respectively.
If $d_e^* = 0$ and $d_f^* = 1$, then $\mathcal{H}_1(ID_e^*) = c_e^* P$ and $\mathcal{H}_1(ID_f^*) = c_f^*(bP)$.
Now, $\mathcal{B}$ calculates and produces the involved:

$$\begin{aligned} e(P,\ \mathcal{V}_f^*) &= e(P_{pub},\ v\prime^* h_4^* \mathcal{H}_1(ID_f^*))e(P_{hlp},\ v\prime^* \mathcal{H}_2(ID_f,\ t*))\mathcal{U}_f^* \\ &= e(P,\ v\prime^* h_4^* c_f^*(xyP) + v\prime^* k^* P_{hlp} + n_f^* P_{pub}). \end{aligned}$$

Implies, $\mathcal{V}_f^* = v\prime^* h_4^* c_f^*(xyP) + v\prime^* k^* P_{hlp} + n_f^* P_{pub}$ and so

$$xyP = (v\prime^* h_4^* c_f^*)^{-1}(\mathcal{V}_f^* - v\prime^* k^* P_{hlp} - n_f^* P_{pub}).$$

This suffices the depiction of Theorem and of $\mathcal{B}$.

Table 2 Efficiency table

Scheme	Key update phase	Key-insulated Agg Proxy Sign Gen Phase	Key-insulated Agg Proxy Sign Ver Phase
IKAPS Scheme	$1\mathcal{T}_m + 2\mathcal{T}_a$ $= 0.002868$ms	$n\mathcal{T}_m + (n-1)\mathcal{T}_a$ $= (0.00159n - 0.001278)$ms	$3\mathcal{T}_p + (2n-2)\mathcal{T}_a$ $= (35.944833 + 0.002556n)$ms

5 Efficiency Analysis

The computational effectiveness of the constructed IKAPS scheme is based on evaluation time of exhausting operations. For this, we take in to account the experimental results carried out by Chen et al. [9], in view of time taken for evaluating different operations as follows: $1\mathcal{T}_p \approx 11.982463$ ms (milli seconds), $1\mathcal{T}_m \approx 0.000312$ ms, $1\mathcal{T}_a \approx 0.001278$ ms. Here a pairing operation symbolized $\mathcal{T}_p$, a scalar multiplication symbolized $\mathcal{T}_m$ in $\mathcal{G}_a$, a point addition symbolized $\mathcal{T}_a$ in $\mathcal{G}_a$. We incorporate these to our constructed scheme as depicted in Table 2.

There are only 3 pairing calculations involved in the key-insulated aggregate proxy signature verification phase of the constructed IKAPS scheme and is a constant irrelevant to the number of proxy signers participate in signing. Also, the communication overhead of the constructed IKAPS scheme is $|\mathcal{G}_a| + |\mathcal{G}_m| = 256$ bytes, i.e., length of the signature is constant.

6 Conclusion

To shield a signature scheme from diverse attacks, one needs to keep securely private keys of the system. To evade harm by key disclosure problem in aggregate proxy signature schemes, we constructed the first efficient IKAPS scheme using pairings in this article. Further, the security of the constructed IKAPS scheme is attained without Forking lemma and so gives tight reductions to the CDH problem.

References

1. Diffie W, Hellman M (1976) New directions in cryptography. IEEE Trans Inf Theory IT 22(6): 644–654
2. Shamir A (1985) Identity-based cryptosystems and signature schemes. In: Blakley GR, Chaum D (eds.) Advances in Cryptology-CRYPTO 1984, LNCS, vol. 196. Springer-Verlag, Berlin Heidelberg, pp 47–53
3. Sahu RA, Padhye S (2011) ID-based signature schemes from bilinear pairing: A Survey. Front Electr Electron Eng China 6(4):487–500
4. Gopal PVSSN, Vasudeva Reddy P, Gowri T (2012) New identity based signature scheme using bilinear pairings over elliptic curves. 3rd IEEE IACC-2013, pp 362–367

5. Dodis Y, Katz J, Xu S, Yung M (2002) Key-insulated public key cryptosystems. In: Knudsen LR (ed.) EUROCRYPT 2002, LNCS, vol. 2332. Springer-Verlag, Berlin Heidelberg, pp 65–82
6. Zhou Y, Cao Z, Chai Z (2006) Identity based key-insulated signature. In: Chen K et al. (eds.) ISPEC 2006, LNCS, vol. 3903. Springer-Verlag, Berlin Heidelberg, pp 226–234
7. Vasudeva Reddy P, Gopal PVSSN (2017) Identity-based key-insulated aggregate signature scheme. J King Saud Univ Comput Inf Sci 29: 303–310
8. Gopal PVSSN., Vasudeva Reddy P (2015) Efficient ID-based key-insulated signature scheme with batch verifications using bilinear pairings over elliptic curves. J Discret Math Sci Cryptogr 18(4): 385–402
9. Chen Y, Yao T, Ren H, Gan Z (2022) Unidirectional identity-based proxy re-signature with key insulation in EHR sharing system. Comput Model Eng Sci 131(3):1497–1513
10. Boneh D, Gentry C, Lynn B, Shacham H (2003) Aggregate and verifiably encrypted signatures from bilinear maps. In: Bihan E (eds.) EUROCRYPT 2003, LNCS, vol. 2656. Springer-Verlag, Berlin Heidelberg, pp 416–432
11. Mambo M, Usuda K, Okamoto E (1996) Proxy signatures: Delegation of the power to sign messages. IEICE Trans Fundam Electron Commun Comput Sci E79-A (9): 1338–1354
12. Zhang F, Kim K (2003) Efficient ID-based blind signature and proxy signature from bilinear pairings. In: Proceedings of the 8th Australasia Conference on INFORMATION SECURITY AND PRIVACY (ACISP'03), vol. 2727. pp 312–323
13. Tiwari N, Padhye S (2013) Analysis on the generalization of proxy signature. Secur Commun Netw 6(5):549–566
14. Wan Z, Lai X, Weng J, Liu S, Hong X (2009) Identity-based key-insulated proxy signature. J Electron 26(6):853–858
15. Pointcheval D, Stern J (2000) Security arguments for digital signatures and blind signatures. J Cryptol 13(3):361–396
16. Lin YC, Wu TC, Tsai JL (2013) ID-Based aggregate proxy signature scheme realizing warrant-based delegation. J Inf Sci Eng 29:441–457
17. Goh EJ, Jarecki S (2003) A signature scheme as secure as the Diffie-Hellman problem. In: Bihan E (ed) EUROCRYPT 2003, LNCS, vol. 2656. Springer-Verlag, Berlin Heidelberg, pp 401–415
18. Katz J, Wang N (2003) Efficiency improvements for signature schemes with tight security reductions. In: Proceedings of the 10th ACM Conference on COMPUTER AND COMMUNICATIONS SECURITY 2003. ACM Digital Library, pp 155–164