Available online at http://scik.org
J. Math. Comput. Sci. 2024, 14:7
https://doi.org/10.28919/jmcs/8489
ISSN: 1927-5307

EFFICIENT AND SECURE CERTIFICATELESS KEY–INSULATED PROXY SIGNATURE SCHEME

MAHESH PALAKOLLU[1], GOWRI THUMBUR[2], P. VASUDEVA REDDY[1,*]

[1]Department of Engineering Mathematics, Andhra University, Visakhapatnam, 530003, India

[2]Department of Electronics and Communication Engineering, GITAM University, Visakhapatnam, 530045, India

Abstract: A proxy signature scheme is a kind of digital signature which permits the original signer to delegate his or her signing capabilities to a proxy signer who can sign on the original signer's behalf. The security of any cryptographic scheme based on the secrecy of private key and the exposure of such keys may result in disastrous circumstances in the communication network of the system. A proxy key insulated scheme was devised to mitigate the impact of private key exposure in proxy signature schemes. In this research, we offer a novel certificateless key-insulated proxy signature technique (CL-KIPS) that employs elliptic curve cryptography on a finite field. This approach reduces the damage caused by private key exposure in any proxy signature scheme. Even if a secret key is released for a limited time, the suggested key insulation technique refreshes the key and prevents the adversary from acquiring the secret from the device for subsequent time periods. The suggested CL-KIPS scheme's security is shown in the ROM model with the premise that the ECDLP is hard. In terms of computing and communication, we contrast our CL-KIPS system with similar and contemporary technologies. The comparison of the findings indicates that the suggested method is suitable for real-world applications like WSNs, VANETs, IoT applications, etc. where the available processing power, bandwidth, and storage capacity are constrained.

Keywords: certificateless cryptography; key-insulation; proxy signature; random oracle model; elliptic curve

---

*Corresponding author
E-mail address: vasucrypto@andhrauniversity.edu.in
Received February 10, 2024

discrete logarithmic problem; resource constrained devices.

2020 AMS Subject Classification: 68M25.

1. INTRODUCTION

Digital signatures play an important role in ensuring data integrity, authentication and non-repudiation for digital communication. In a digital signature scheme, messages are signed by the corresponding signer's public key. Diffe and Hellman (1976) [1] introduced the the concept of Public Key Cryptography (PKC) in which each user has a public key and private key. In PKC, a signer can sign a document using his private key and any user can verify the validity of the signature using the public of the signer. In multiuser environment, the authentication, revocation, storage of public keys leads to lot of key management problems. To eliminate the burden of certificate management in traditional PKC, in 1984 Shamir [2] proposed an idea of identity based cryptography (IBC) in which a PKG generates the private key of the users. However, key escrow problem is an inherent problem in IBC. Al-Riyami and Paterson [3] introduced a novel architecture called certificateless public key cryptography (CL-PKC) in 2003 to eliminate problem of key-escrow in IBC and key management problems in PKC. In CL-PKC, the user's private key is divided into two parts. PKG generates the partial private key, and the user chooses the secret key. Since the PKG has no control over a user's private key, the key- escrow issue can be resolved.

The concept of proxy signature was initially proposed in 1996 by Mambo et al. [4] and involved three entities: an original signer, a proxy signer, and a verifier. In a proxy signature scheme, the original signer delegates his/her capabilities to a proxy signer to sign documents on their behalf. Many cryptographic schemes have been devised based on the secrecy of signing keys, i.e., If the secret keys are exposed then the security of the entire will be lost. To overcome this problem, in 2002, Dodis et al. [5] introduced a key insulated mechanism (KIS). Many KIS schemes have been reported in literature with PKI and other cryptographic frameworks [6,7]. The idea behind the KIS mechanism is dividing the master secret key for discrete time periods and user can update the temporary signing key by adding the helper secret key to the current time period signing key. This private key can update by the user periodically. However, the public key associated with this signing key remains same for the entire life period.

In 2009, Wan et al. [8] created the first Identity Based key insulated proxy signature strategy utilizing bilinear pairings in an effort to reduce the harm caused by the exposure of the proxy signing key in a proxy signature scheme. This strategy, which makes use of the most costly pairing operations, has been shown to be secure in the random oracle model (ROM) while also being computationally inexpensive. To adopt the scheme in resource constrained devices like WSNs, IoT applications [9], it requires less computation cost, lower bandwidth and higher memory. Also, the cryptographic scheme having larger key sizes requires high computational cost, huge memory space and high bandwidth, which results less efficiency.

Due to these limitations, the design of schemes with smaller keys in size is more attractive for resource constrained environments. Elliptic curve cryptography (ECC) independently developed by Koblitz and Miller [10] plays a vital role in the design of lightweight cryptographic schemes which provides higher level of security with smaller keys [11,12]. For various purposes, bilinear pairings over elliptic curves have been used to develop a number of cryptographic techniques based on the ECC [13, 14, 15, 16, 17]. However, pairing based cryptographic systems are not very efficient in their implementation due to the significant computational cost required in the assessment of pairing operations and map to point hash functions. Hence cryptographic schemes based on ECC without bilinear pairings are more efficient.

Related work

Many key-insulated signature (KIS) techniques and proxy signature schemes have been reported in the literature separately [5, 6, 7, 8, 14, 18, 19, 20, 21, 22, 23, 24]. However, the literature has very few key-insulated proxy signature techniques. Hong et al. [20] suggested a secure PKI-based, key-insulated proxy signing system for mobile agents in 2007. The security of this approach is proved in the random Oracle model. Wan et al. [8] introduced the first identity-based key-insulated proxy signature system in 2009, The authors demonstrated that the suggested technique is a strong and perfect key-insulated signature scheme that is unforgeable and is proved in the random oracle model. In 2011, Chen [25] proposed an ID-based KIS system in the standard model. In 2019, Chen [26] designed an ID-based KIS proxy signature scheme in standard model. In 2020, Chen [27] suggested an ID-based parallel key insulated proxy signature solution for random oracles. Until yet, these are the only key-insulated proxy signature techniques published in the literature, and they all rely on ID-based cryptographic parameters. However, there is no key-insulated proxy signature technique in the certificateless framework [28]. Furthermore, all of the preceding techniques employ bilinear pairings over elliptic curves, a costly cryptographic procedure. As a result, in this work, we offer a novel key-insulated proxy signature strategy in a certificateless scenario that does not need bilinear pairings.

1.1. Our contributions

Inspired by the issues mentioned above, we design a new and secure lightweight Certificateless Key Insulated Proxy Signature Scheme (CL-KIPS) using elliptic curve cryptography. The main contributions of the paper are summarized as follows:

i) We presented a lightweight Certificateless Key Insulated Proxy Signature Scheme (CL-KIPS) using elliptic curve cryptography. This scheme combines the concepts of key insulated mechanism and proxy signature in CL-based framework.

ii) Our CL-KIPS scheme is proven secure in the random oracle model (ROM) under the hardness ECDLP problem.

iii) The performance analysis of our scheme shows that the Computational and communicational efficiency is much better than the existing schemes.

1.2. Organization

The remaining part of the paper is organized as follows. Section 2 outlines some preliminaries. Section 3 gives the syntax of our CL-KIPS scheme. Section 4 gives the proposed CL-KIPS scheme. Section 5 gives the security arguments of our scheme. Section 6 presents performance analysis of our scheme. Section 7 provides the summary of this paper.

2. PRELIMINARIES

In the following we present mathematical and cryptographic assumptions related to elliptic curve.

2.1. Elliptic Curve Group

In ECC, an elliptic curve group $E\left(F_p\right)$ is considered over F_p, where $p>3$ is a prime, as $y^2=(x^3+ax+b)$, $a,b\in F_p$ and $4a^3+27b^2\neq 0.$ The set $G=\{(x,y)/(x,y)\in E\}\cup O$ is an abelian group with the chord-and-tangent rule [10,11]. Let $\langle P\rangle$ be generator of the elliptic curve group $G.$ For more details elliptic curve group can be found in [10, 11]. Let P be the generator of $G,$ and the order of G is q. Let $k\in Z_q^*.$ The scalar multiplication is defined as $kP=P+P+........P(k\ \textbf{times})$.

2.2. Elliptic Curve Discrete Logarithm Problem (ECDLP)

Suppose that for a given $P,Q\in G,$ the ECDLP is to compute $x\in Z_q^*,$ such that $Q=xP.$

Computation of x from P and Q is computationally hard by any polynomial-time bounded algorithm.

2.3. Notations

The following TABLE 1 provides the notations and their meanings..

Table 1: Notations and their Meanings

Notation	***Meaning***
KGC	Key Generation Centre
G	Cyclic group of prime order q.
$params$	System Parameter.
msk, P_{Pub}	Master Secret Key, Master Public Key
PPK	Partial Private Key
PK_{ID_i}, SK_{ID_i}	Public Key and Secret Key of the User *i*
hsk	Helper Secret Key
$UHK_{B,t}$	Updated Helper Key
σ	Signature on a message m.
Ω	Key Insulated Proxy Signature
$IPSK_B$	Initial Proxy signing Key

$TSK_{B,t}$	Temporary proxy signing key for time period t
Hi	Cryptographic one way hash functions
Adv_1, Adv_2	Type-I and Type-II adversaries
ξ	Challenger
ECDLP	Elliptic Curve Discrete Logarithm Problem

3. FRAMEWORK OF CL-KIPS SCHEME

A CL-KIPS involves ten entities: the Key Generation Centre (KGC), the original signer, proxy signer, helper and the verifier. The proposed CL-KIPS scheme designed with the following ten algorithms. The detailed description of each of these algorithms is as follows.

1) **Setup:** KGC performs the Setup algorithm by taking the security parameter $k \in Z^+$ as input and outputs the common system parameters *params* and master secret key msk. KGC publishes *params* and keeps master secret key (msk) secretly.

2) **Partial Private Key Generation:** KGC performs this algorithm to generate PPK of a particular user and sends to user via a secure channel.

3) **User Key Generation:** User performs this probabilistic algorithm by taking system parameters *params*, his identity $ID \in \{0,1\}^*$ and corresponding partial private key D_{ID} as inputs and choose $x_{ID} \in Z_q^*$ at random to compute $X_{ID} = x_{ID}P.$ User sets x_{ID} as his secret value and sets the users public and private key pair (PK_{ID}, SK_{ID}).

4) **Delegation Generation:** Taking *params*, master public key, an original signers identity ID_A with its private key D_A, a warrant m_w as input, original signer runs this algorithm and generates the delegation $\sigma_{A \to B}$ on the warrant m_w.

5) **Delegation Verification:** Given a delegation $\sigma_{A \to B}$ on the warrant m_w, the proxy signer verifies and accepts the delegation of original signer if the delegation is valid, rejects otherwise.

6) **Proxy Initial Key Generation:** The proxy signer executes this algorithm to compute initial proxy signing key TSK_{ID} and helper secret key.

7) **Helper Update:** Helper performs this algorithm to update helper key as $UHK_{ID,t,t-1}$.

8) **User Key Update:** User performs this algorithm with the inputs signing key $TSK_{ID,t-1}$ for the time period $t-1$, updated helper key $UHK_{ID,t,t-1}$ for the time period indices $t, t-1$, and computes user's temporary signing key $TSK_{ID,t}$ for the current time period t.

9) **Proxy Signature Generation:** Proxy signer performs this algorithm with the inputs system parameters, signers identity $ID \in \{0,1\}^*$, $m \in \{0,1\}^*$ and proxy signer's updated private key TSK_{ID} and produces a proxy signature Ω_{ID}.

10) **Proxy Signature verification:** Any verifier performs this algorithm by taking (m, Ω_{ID}) with signers identity $ID \in \{0,1\}^*$ and corresponding public key PK_{ID}, system parameters *params* as input and outputs '1' if Ω_{ID} is a valid sign on message $m \in \{0,1\}^*$ or '0' otherwise.

4. PROPOSED PAIRING-FREE CL-KIPS SCHEME

Our Certificateless key insulated proxy signature scheme (CLKIPS) consists of ten algorithms, the detailed description of each of these algorithms are presented as follows:

1) **Setup:** For a given security parameter $k \in Z^+$, KGC performs the following steps to produce the system parameters and the master key.

i) KGC selects an additive group G of elliptic curve points whose order is a prime q and P is its generator.

ii) KGC randomly selects the master secret key $msk = s \in Z_q^*$ and calculates the system master public key $P_{pub} = sP$.

iii) KGC selects cryptographic hash functions $H_i:\{0,1\}^* \to Z_q^*$ for $i=1,2,3,4,5$ and $H_i':\{0,1\}^* \to Z_q^*$ for $i=2,3.$

iv) KGC publishes $\text{params}=\{q,G,P,P_{pub},H_i,H_i'\}$ as the system parameters and secretly keeps the master secret key $msk=s.$

2) **Partial Private Key Generation:** Upon receiving user's identity $ID_i, i\in\{A,B\}$, the KGC executes this algorithm as given below to generate Partial private key.

i) KGC selects $r_{ID_i}\in Z_q^*$ and calculates $R_{ID_i}=r_{ID_i}P.$

ii) KGC computes $d_{ID_i}=r_{ID_i}+sh_1 \bmod q.$ Where $h_1=H_1\left(ID_i,R_{ID_i},P_{pub}\right).$

iii) KGC sends the partial private key as $D_{ID_i}=\left(d_{ID_i},R_{ID_i}\right)$ to the user ID_i securely. The user ID_i verifies $d_{ID_i}P=R_{ID_i}+h_1P_{pub}$ to verify the validity of $D_{ID_i}.$

3) **User Key Generation:** A user ID_i, chooses $x_{ID_i}\in Z_q^*$ and computes $X_{ID_i}=x_{ID_i}P.$ Set $PK_{ID_i}=\left(X_{ID_i},R_{ID_i}\right)$ as public key and $SK_{ID_i}=\left(d_{ID_i},x_{ID_i}\right)$ as secret key.

4) **Delegation Generation:** The original signer A with the identity ID_A prepares a warrant m_w and delegates his or her signing rights to the proxy signer B whose identity is $ID_B.$

i) The original signer ID_A chooses $k_A\in Z_q^*$ and computes $K_A=k_AP.$

ii) Computes $h_2=H_2\left(m_w,K_A,R_{ID_A},PK_{ID_A}\right),\ h_3=H_3\left(ID_A,R_{ID_A},ID_B,PK_{ID_B},P_{pub}\right).$

iii) Computes $\sigma_w=k_A+h_2d_{ID_A}++h_3x_{ID_A} \bmod q.$

iv) The original signer A sends $\left(m_w,K_A,R_{ID_A},\sigma_w\right)$ to the proxy signer $B.$

5) **Delegation Verification:** Upon receiving the $\left(m_w,K_A,R_{ID_A},\sigma_w\right)$, the proxy signer B verifies that

$$\sigma_wP=K_A+h_2R_{ID_A}+h_3X_A+h_1P_{pub};\ h_2=H_2\left(m_w,K_A,R_{ID_A},PK_{ID_A}\right),$$

$h_3 = H_3\left(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub}\right)$, $h_1 = H_1\left(ID_A, R_{ID_A}, P_{pub}\right)$.

If the equation holds, B accepts that delegation was correct. Otherwise B rejects the delegation

6) **Proxy Initial Key Generation:** After accepting the delegation the proxy signer B generates the initial proxy signing key as follows.

i) Proxy signer B chooses $h_{SK_B} \in Z_q^*$ and computes $V_B = h_{SK_B} P$.

ii) Proxy signer B computes Initial proxy signing key

$IPSK_{B,0} = d_{ID_B} + x_{ID_B} H_2'(m_w, \sigma_w, ID_A, ID_B, PK_B) + h_{SK_B} H_3'(ID_B, V_B, 0)$. B returns the initial proxy signing key as $\left(IPSK_{B,0}, V_B, R_{ID_A}\right)$ and send the helper secret key h_{SK_B} to the helper B.

7) **Helper Update:** On input of $params$, time period indices $(t, t-1)$, the helper for the user ID_B works as follows to update his key.

i) Compute $V_B = h_{SK_B} P$.

ii) Computes $UHK_{B,t} = h_{SK_B}\left[H_3'(ID_B, V_B, t) - H_3'(ID_B, V_B, t-1)\right]$. Return the updated helper key $UHK_{B,t}$ to the proxy signer B.

8) **User Key Update:** On input of $params$, time period t, updated helper key $UHK_{B,t}$ and the $IPSK_{B,t-1}$; the proxy signer ID_B performs the following.

i) Compute $IPSK_{B,t} = IPSK_{B,t-1} + UHK_{B,t-1}$.

ii) Return the temporary proxy signing key for the time period t as $TSK_{B,t} = \left(IPSK_{B,t}, V_B, R_{ID_A}\right)$. Note that at the time period t, $IPSK_{B,t}$ is always set to be

$$
\begin{aligned}
IPSK_{B,t} = {} & d_{ID_B} + x_{ID} H_2'(m_w, \sigma_w, ID_A, ID_B, PK_B) \\
& + h_{SK_B} H_3'(ID_B, V_B, t).
\end{aligned}
$$

9) **Proxy Signature Generation:** For a given time period t, a message m, the proxy signer ID_B generates the signature using a temporary proxy signing key $IPSK_{B,t}$. B do the following.

i) Chooses $u_B \in_R Z_q^*$ and computes $U_B = u_B P$.

ii) Calculate $h_4 = H_4\left(t, m, m_w, R_{ID_A}, U_B\right)$, $h_5 = H_5\left(t, m, m_w, R_{ID_A}, U_B\right)$.

iii) Calculate $\sigma = h_4\ IPSK_{B,t} + u_B\ h_5 \bmod q$.

iv) Output a CL-Key Insulated Proxy signature as $\Omega = \left(R_{ID_A}, V_B, U_B, K_A, \sigma, m, m_w\right)$ and sends it to a verifier.

10) **Proxy Signature Verification:** Upon receiving $params$, t, ID_A, ID_B, PK_{ID_B}, R_{ID_A}, U_B, a message signature pair (m, Ω), helper public key V_B, any verifier verifies the signature as follows.

i) Computes $h_1 = H_1\left(ID_A, R_{ID_A}, P_{pub}\right)$, $h_2 = H_2\left(m_w, K_A, R_{ID_A}, PK_{ID_A}\right)$,

$h_3 = H_3\left(ID_A, R_{ID_A}, ID_B, PK_{ID_B}, P_{pub}\right)$, $h_4 = H_4\left(t, m, m_w, R_{ID_A}, U_B\right)$, $h_5 = H_5\left(t, m, m_w, R_{ID_A}, U_B\right)$.

$h_2' = H_2'\left(m_w, \sigma_w, ID_A, ID_B, PK_B\right), h_3' = H_3'\left(ID_B, V_B, t\right)$.

ii) Verify whether the equation $\sigma P - h_5 U_B = h_4\ \left(R_{ID_B} + h_1 P_{pub} + h_2' X_B + h_3' V_B\right)$.

5. SECURITY OF OUR CL-KIPS

The security of the proposed CL-KIPS scheme can be captured by the following security games against the two types of adversaries [28].

Theorem 1: Under the ECDLP assumption and in the random oracle model, suggested PF-CLKIPS Scheme is completely key insulated and unforgeable against a Type-I adversary.

Proof: Consider a Type-I adversary Adv_1 who attempts to break the security of the proposed signature scheme with a non-negligible probability. The following will demonstrate how to create an additional algorithm ξ that, with the adversary's assistance, can solve the ECDLP. The challenger's ξ aim is to compute $s \in Z_q^*$ from the given instance of the ECDLP. For this ξ takes ID^* as the target identity.

- ***Initialization Phase:*** Challenger ξ sets $P_{pub} = Q = sP$, and executes the setup algorithm to outputs public parameters and master secret key s to Adv_1. ξ keeps s secretly.

- ***Queries Phase:*** Adv_1 makes a series of queries and ξ answers these queries as follows:
  - ***Queries on Oracle*** $\mathrm{H}_1 : H_1(ID_i, R_i, P_{pub})$: ξ keeps an empty list L_1 of with the tuples of the form $(ID_i, R_i, P_{pub}, l_{1i})$. when Adv_1 queries $H_1\,(ID_i, R_i, P_{pub})$, ξ will determine whether or not the tuple $(ID_i, R_i, P_{pub}, l_{1i})$ is present in the list. Returns l_{1i}, if it is present in; if not, selects a random value l_{1i}, and inserts it before returning to Adv_1.
  - ***Queries on oracle*** H_2: ξ maintains an empty list L_2 list of tuple $(m_w, K_A, R_{ID_A}, PK_{ID_A}, l_{2i})$. When Adv_1 queries H_2 $(m_w, K_A, R_{ID_A}, PK_{ID_A})$, ξ will check whether the tuple $(m_w, K_A, R_{ID_A}, PK_{ID_A}, l_{2i})$ presents in L_2 list or not. If it appears in L_2 then ξ returns l_{2i} otherwise ξ picks random l_{2i} and add to L_2. Finally ξ outputs l_{2i}.
  - ***Queries on oracle*** H_3: $H_3(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub})$: ξ maintains an empty list L_3, list of tuple $(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub}, l_{3i})$. When Adv_1 queries H_3 $(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub})$, ξ checks whether the tuple $(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub}, l_{3i})$ is in the list L_3 or not. If exists in then ξ outputs l_{3i}. Else, ξ choses a random l_{3i} and inserts to L_3. Finally ξ returns l_{3i} to Adv_1.
  - ***Queries on oracle*** H_2':$H_2'(m_w, \sigma_w, ID_A, ID_B, PK_B)$: ξ maintains an empty list L_2' list of tuple $(m_w, \sigma_w, ID_A, ID_B, PK_B, l_{2i}')$. When Adv_1 queries H_2' $(m_w, \sigma_w, ID_A, ID_B, PK_B)$, ξ will check

whether the tuple $(m_w, \sigma_w, ID_A, ID_B, PK_B, l'_{2i})$ is in the L'_2 list or not. If it presents in L'_2 then ξ outputs l'_{2i}. Else, ξ chooses a random l'_{2i} and inserts to L'_2. Finally ξ returns l'_{2i} to Adv_1.

- ***Queries on oracle*** H'_3**:**$H'_3(ID_i, V_i, t_i)$**:** ξ maintains an empty list L'_3 list of tuple $(ID_i, V_i, t_i, l'_{3i})$ When Adv_1 queries H'_3 (ID_i, V_i, t_i), ξ will check whether the tuple $(ID_i, V_i, t_i, l'_{3i})$ is in the list L'_3 or not. If it exists, ξ returns l'_{3i}. Else, ξ picks a random l'_{3i} and adds to L'_3. Finally ξ returns l'_{3i} to Adv_1.
- ***Queries on oracle*** H_4**:**$H_4(t, m, m_w, R_{ID_A}, U_B)$**:** ξ maintains an empty list L_4, list of tuple $(t, m, m_w, R_{ID_A}, U_B, l_{4i})$. When Adv_1 makes a query on $(t, m, m_w, R_{ID_A}, U_B)$, ξ looks the list L_4 to check whether the tuple $(t, m, m_w, R_{ID_A}, U_B, l_{4i})$ exists in the list L_4 or not. If it exists then ξ returns l_{4i}. Otherwise, ξ chooses a random l_{4i} and inserts to L_4. Finally, ξ sends l_{4i} to Adv_1.
- ***Queries on oracle*** H_5**:**$H_5(t, m, m_w, R_{ID_A}, U_B)$**:** ξ maintains an empty list L_5, list of tuple $(t, m, m_w, R_{ID_A}, U_B, l_{5i})$. When Adv_1 makes a query on $(t, m, m_w, R_{ID_A}, U_B)$, ξ will check whether the tuple $(t, m, m_w, R_{ID_A}, U_B, l_{5i})$ exists in L_5 list or not. If it exists, then ξ returns l_{5i}. Otherwise, ξ picks a random l_{5i} and adds to L_5. Finally ξ returns l_{5i}.
- ***Reveal Partial Secret key Oracle*** $(PSK(ID_i))$**:** When Adv_1 makes a query on $PSK(ID_i)$, ξ ξ maintains an initial empty list L_{psk} of tuple $(ID_i, \mathrm{d}_i, \mathrm{R}_i)$ and returns d_i, whether this question has already been asked. Otherwise if $ID_i \neq ID^*$, ξ chooses $a_i \in Z_q^*$ and sets $d_i = a_i$ and adds $(ID_i, \mathrm{d}_i, \mathrm{R}_i)$ to L_{psk} and returns d_i. If $ID_i = ID^*$, ξ aborts.

- ***Create User Oracle*** $(Cuser(ID_i))$**:** To answer this query ξ maintains an empty list L_{Cuser} with the tuple of the form (ID_i, x_i, PK_i). When Adv_1 makes a query on $Cuser(ID_i)$, ξ looks the list L_{Cuser} and outputs PK_i whether this question has already been asked. Otherwise ξ performs as follows.

  (i) If $ID_i \neq ID^*$, ξ selects $a_i, b_i, x_i \in Z_q^*$ and sets $R_i = a_i P - b_i P_{pub}$, $H_1(ID_i, R_i, P_{pub}) = b_i$ and $X_i = x_i P$. ξ sets $PK_i = (X_i, R_i)$, and inserts $(ID_i, R_i, P_{pub}, b_i)$ to the list L_1 and (ID_i, X_i, PK_i) to the L_{Cuser}. ξ sends PK_i to Adv_1 as a response. Clearly (R_i, X_i, h_{1i}) satisfies $d_i P = R_i + h_{1i} P_{pub}$.

  (ii) If $ID_i = ID^*$, ξ generates $a_i, b_i, x_i \in Z_q^*$ and sets $R_i = a_i P$, $H_1(ID_i, R_i, P_{pub}) = b_i$ and $X_i = x_i P$. ξ sets $PK_i = (X_i, R_i)$ and adds $(ID_i, R_i, P_{pub}, b_i)$ to the list L_1 and (ID_i, x_i, PK_i) to L_{Cuser}. ξ returns PK_i to Adv_1.

- ***Reveal Secret key Oracle*** $(RSK(ID_i))$**:** When Adv_1 queries $RSK(ID_i)$, If $ID_i = ID^*$, ξ stops the game. Otherwise ξ searches (ID_i, x_i, PK_i), (ID_i, d_i, R_i) from L_{Cuser}, L_{PSk} lists respectively and recovers x_i, d_i. ξ sets $SK_i = (d_i, x_i)$ and sends SK_i to Adv_1. If there is no corresponding tuple in L_{Cuser}, L_{PSK}, ξ asks a query on $Cuser(ID_i)$ to produce x_i queries $PSK(ID_i)$ to generate d_i. ξ inserts in L_{Cuser}, L_{PSK} respectively. Finally ξ sends SK_i.

- **Replace Public key Oracle** $(RPK(ID_i))$**:** When Adv_1 queries $RPK(ID_i)$, ξ searches (ID_i, x_i, PK_i) in the list L_{Cuser} and replaces $PK_i = PK_i'$ and $x_i = \perp$.

- ***Queries on Delegation Generation:*** When Adv_1 queries (m_w, ID_A, ID_B), ξ generates $\beta_A, \delta_A \in Z_q^*$, Computes $h_1 = H_1(ID_A, R_A, P_{pub})$, $h_3 = H_3(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub})$, $K_A = \beta_A P - \delta_A(R_A + h_1 P_{pub}) - h_3 X_A$. ξ sets $\sigma_W = \beta_A$ and $h_2 = \delta_A$. At last ξ returns

(K_A, σ_W) to Adv_1 and adds $(m_w, SK_A, RID_A, PK_{ID_A}, \delta_A)$ to list L_2. Note that delegation (K_A, σ_W) generated in this way satisfies the equation: $\sigma_W P = K_A + h_2 R_{ID_A} + h_3 X_A + h_1 P_{pub}$.

- ***Queries on Proxy Key Generation: On receiving a*** query (m_w, ID_A, ID_B) from Adv_1, ξ receives (K_A, σ_W) through Delegation generation queries. If $ID_B = ID^*$, ξ stops simulation. Otherwise, ξ finds the tuples $(ID_B, x_B, PK_{ID_B}), (ID_B, d_B, R_B)$ from L_{Cuser}, L_{PSk} respectively and recovers x_B, d_B. and recovers $(m_w, \sigma_w, ID_A, ID_B, PK_B, l_{2i})$, (ID_B, V_B, t_B, l_{3i}) from L_2', L_3' and also computes the $IPSK_B$ $IPSK_B = d_B + x_B\ l_{2i}' + hsk_B\ l_{3i}'$ by choosing $hsk_B \in Z_q^*$ randomly, ξ returns the proxy key $IPSK_B$ to Adv_1.

- ***Queries on Temporary Signing key Oracle*** $(TSK(ID_i))$**:** When Adv_1 queries $TSK(ID_i)$ for the period t_i, ξ searches the list L_{TSK} and gives TSK_{ID_i,t_i}, if this query already asked. Otherwise ξ do the following.

(i) If $ID_i = ID^*$, ξ stops the game.

(ii) If $ID_i \neq ID^*$, ξ selects $v_i \in Z_q^*$ and sets $V_i = v_i P$ and computes $IPSK_{ID_i,t_i} = h_{4i}[h_{3i}' v_i + h_{2i}' x_i]$,

where $h_{2i}' = H_2'(m_w, \sigma_w, ID_A, ID_B, PK_B)$, $h_{3i}' = H_3'(ID_i, V_i, t_i)$, $h_{4i} = H_4(t, m, m_w, R_{ID_A}, U_B)$.

$\therefore TSK_{ID_i,t_i} = \left[IPSK_{ID_i,t_i}, V_i\right]$, ξ outputs TSK_{ID_i,t_i} as a temporary signing key and returns to Adv_1.

- ***Proxy Signing Oracle:*** When Adv_1 queries (m, m_w, ID_A, ID_B), ξ first asks queries on $H_1, H_2, H_3, H_2', H_3', H_4$ and H_5 for $i = A$ or B and (ID_i, d_i, R_i) from L_{PSK} and (ID_i, x_i, PK_i) from L_{Cuser}.

(i) If $ID_i \neq ID^*$, ξ recovers TSK_{ID_B,t_i}, V_B from L_{TSK} list and set $\sigma_B = IPSK_{ID_B,t_B}$ and $V_B = v_B P$, $U_B = h_{4i}\left[R_i + h_{1i} P_{pub}\right](-h_{5i})^{-1}$. ξ outputs $\Omega_B = (R_{ID_A}, V_B, U_B, \sigma_B, ID_A)$.

(ii) If $ID_i = ID^*$, ξ selects a random $v_B \in Z_q^*$ and computes $V_B = v_B P$ and then calculate $\sigma_B = h'_{3i} h_{4i} v_B$, $U_B = h_{4i}\left[R_i + h_{1i} P_{pub} + h'_{2i} X_i\right](-h_{5i})^{-1}$. now ξ outputs $\Omega_B = \left(R_{ID_A}, V_B, U_B, \sigma_B, ID_A\right)$ as a valid signature.

Now ξ outputs the proxy signature as $\Omega_B = \left(R_{ID_A}, V_B, U_B, \sigma_B, ID_A\right)$ note that (m, m_w, Ω_B) is a valid signature. $\sigma P - h_5 U_B = h_4\left(R_{ID_A} + h_1 P_{pub} + h_2 X_{ID_B} + h_3 V_B\right)$.

- ***Forgery/Output:*** Finally, Adv_1 returns a valid forged signature tuple $\left(t_i^*, m^*, \mathrm{m}_\omega^*, \Omega_B^*\right)$ as its forgery where $\Omega_B^* = \left(R_{ID_A}^*, V_B^*, U_B^*, \sigma^*, ID_A^*\right)$. If $ID_i \neq ID^*$, ξ aborts. Otherwise ξ do the following.

Let $\Omega_B^{*(1)} = \left(R_{ID_A}^*, V_B^*, U_B^*, \sigma^{*(1)}, ID_A^*\right)$. By Forking Lemma [29], if ξ repeats the same with random tape and with different hash values $H_1, H_2, H_3, H'_2, H'_3, H_4, H_5$. Adv_1 will output another four signatures $\Omega_B^{*(j)} = \left(R_{ID_A}^*, V_B^*, U_B^*, \sigma^{*(j)}, ID_A^*\right)$ for $j = 2,3,4,5$ and the following equation holds.

$$\sigma^{*(j)} P - h_5^{*(j)} U_B^* = h_4^{*(j)}\left(R_{ID_B}^* + h_1^{*(j)} P_{pub} + h_2^{*(j)} X_B + h_3^{*(j)} V_B^*\right) \quad \text{for } j = 1,2,3,4,5. \tag{1}$$

By u_B, r_i, s, x_i, v_B we denote discrete logarithms of $U_B, R_{ID_A}, P_{pub}, X_B$ and V_B respectively, i.e. $U_B = u_B P$, $R_{ID_A} = r_i P$, $P_{pub} = sP$, $X_B = x_i P$ and $V_B = v_B P$.

From(1) we get, $\sigma^{*(j)} - h_5^{*(j)} u_B^* = h_4^{*(j)}\left(r_i^* + h_1^{*(j)} s + h_2^{*(j)} x_i^* + h_3^{*(j)} v_B^*\right)$, for $j = 1,2,3,4,5$. (2)

Finally, ξ solves the unknowns $r_i^*, u_B^*, s, x_i^*, v_B^*$ by solving these linearly independent equations (2) and outputs ‘ s ’ as the solution of ECDLP.

Theorem 2: Under the ECDLP assumption, our PF-CLKIPS scheme is perfectly key insulated and unforgeable against a Type-II adversary Adv_2 in the Random oracle model.

Proof: Assume that Adv_2 be a Type-II adversary who break the proposed PF-CLKIPS scheme with non-negligible probability. We will now demonstrate how to create an another algorithm ξ, with the assistance of the adversary, can solve the ECDLP.

Initialization Phase: Challenger ξ sets $P_{pub} = sP$ and runs the setup algorithm to output *params, msk* o the adversary Adv_2

- ***Queries Phase:*** Adv_2 makes a series of queries and ξ can answer these queries as follows:
  - ***Queries on Oracle*** H_1: Upon receiving a H_1 query on (ID_i, R_i, P_{pub}), ξ searches the list L_1 for the thple $(ID_i, R_i, P_{pub}, l_{1i})$. ξ returns l_{1i}, if it appears in L_1. Otherwise, ξ selects a random l_{1i} and sends it to Adv_2. Finally, l_{1i} inserts to the list L_1.

- ***Queries on oracle*** H_2: ξ maintains an empty list L_2 list of tuple $(m_w, K_A, R_{ID_A}, PK_{ID_A}, l_{2i})$. When Adv_2 queries H_2 $(m_w, K_A, R_{ID_A}, PK_{ID_A})$, ξ searches the list L_2 for the tuple $(m_w, K_A, R_{ID_A}, PK_{ID_A}, l_{2i})$. If it appears in L_2 then ξ returns l_{2i} Adv_2. Otherwise, ξ selects a random l_{2i} and adds to the list L_2. Finally ξ returns l_{2i}.

- ***Queries on oracle*** H_3: ξ maintains an empty list L_3, list of tuple $(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub}, l_{3i})$. When Adv_2 queries H_3 $(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub})$, ξ will check whether the tuple $(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub}, l_{3i})$ exists or not. If it exists in L_3, then ξ returns l_{3i}. Otherwise, ξ chooses a random l_{3i} and inserts to L_3. Finally ξ returns l_{3i} to Adv_2.

- ***Queries on oracle*** H_2': $H_2'(m_w, \sigma_w, ID_A, ID_B, PK_B)$: ξ keeps an initial empty list L_2' and has the tuples of the form $(m_w, \sigma_w, ID_A, ID_B, PK_B, l_{2i}')$. When Adv_2 performs a H_2' query with the tuple $(m_w, \sigma_w, ID_A, ID_B, PK_B)$, ξ will check whether the tuple $(m_w, \sigma_w, ID_A, ID_B, PK_B, l_{2i}')$ is exists or not. If this tuple exists in L_2' then ξ returns l_{2i}'. Otherwise, ξ chooses l_{2i}' and inserts to the list L_2'. Finally ξ sends l_{2i}' to Adv_2.

- ***Queries on oracle*** H_3'**:**$H_3'(ID_i, V_i, t_i)$**:** ξ maintains an initially empty list L_3', $(ID_i, V_i, t_i, l_{3i}')$ When Adv_2 asks a H_3' query on (ID_i, V_i, t_i), ξ will checks for the tuple $(ID_i, V_i, t_i, l_{3i}')$ in L_3'. If it is in L_3' then ξ gives l_{3i}'. Otherwise, ξ chooses a random l_{3i}' and inserts to L_3'. Finally ξ returns l_{3i}' to the Adv_2.

- ***Queries on oracle*** H_4**:** $H_4(t, m, m_w, R_{ID_A}, U_B)$**:** ξ has an initially empty list L_4, and has tuple of the form $(t, m, m_w, R_{ID_A}, U_B, l_{4i})$. When Adv_2 makes a query on $(t, m, m_w, R_{ID_A}, U_B)$, ξ will check whether the tuple $(t, m, m_w, R_{ID_A}, U_B, l_{4i})$ is in the list or not. If it exists in L_4 then ξ returns l_{4i}. Otherwise, ξ chooses a random l_{4i} and inserts to the list L_4. Finally ξ returns l_{4i}.

- ***Queries on oracle*** H_5**:** $H_5(t, m, m_w, R_{ID_A}, U_B)$**:** ξ maintains an initially empty list L_5, and has tuple of the form $(t, m, m_w, R_{ID_A}, U_B, l_{5i})$. When Adv_2 makes a query on $(t, m, m_w, R_{ID_A}, U_B)$, ξ will searches for the tuple $(t, m, m_w, R_{ID_A}, U_B, l_{5i})$ in the list L_5. If such tuple exists in L_5 then ξ gives l_{5i}. Otherwise, ξ chooses a random l_{5i} and inserts to L_5. Finally ξ outputs l_{5i} to Adv_2.

- ***Create User Oracle*** $(Cuser(ID_i))$**:** ξ keeps an initially empty list L_{Cuser} with the tuples of the form (ID_i, x_i, PK_i). when Adv_2 asks a query on $Cuser(ID_i)$, ξ searches the list L_{Cuser} and returns PK_i if such entry already in the list L_{Cuser}. Otherwise ξ does as follows.

  (i) If $ID_i \neq ID^*$, then the algorithm ξ selects $a_i, b_i, x_i \in Z_q^*$ and sets $R_i = a_i \mathrm{P} - b_i P_{pub}$, $H_1(ID_i, R_i, P_{pub}) = b_i$ and $X_i = x_i \mathrm{P}$. ξ sets $PK_i = (X_i, R_i)$, and inserts $(ID_i, R_i, P_{pub}, b_i)$ in L_1 and (ID_i, X_i, PK_i) to the L_{Cuser}. Finally, ξ returns PK_i to Adv_2.

  (ii) If $ID_i = ID^*$, ξ generates $a_i \in Z_q^*$ and sets $R_i = a_i \mathrm{P}$, $H_1(ID_i, R_i, P_{pub}) = h_{1i}$ and $X_i = Q = \alpha \mathrm{P}$, ξ sets $PK_i = (X_i, R_i)$ and adds $(ID_i, R_i, P_{pub}, h_{1i})$ to the list L_1 and $(ID_i, \perp, PK_i)$ to L_{Cuser}. ξ returns PK_i to Adv_2 as a response.

- ***Reveal Secret key Oracle*** $(RSK(ID_i))$: When Adv_2 makes a query on $RSK(ID_i)$, If $ID_i = ID^*$, ξ aborts the simulation. Otherwise if $ID_i \neq ID^*$, ξ finds the tuple (ID_i, x_i, PK_i) in L_{Cuser} and recovers x_i, d_i. ξ sets $SK_i = (d_i, x_i)$ and gives SK_i to Adv_2. If not ξ performs a query on $Cuser(ID_i)$ to generate x_i. Here d_i is known to Adv_2. ξ iserts in L_{Cuser}. Finally, ξ returns SK_i.
- ***Queries on Delegation Generation:*** When Adv_2 makes a delegation query on the tuple (m_w, ID_A, ID_B), ξ chooses $\beta_A, \delta_A \in Z_q^*$, and computes $h_1 = H_1(ID_A, R_A, P_{pub})$, $h_3 = H_3(ID_A, ID_B, R_{ID_A}, PK_{ID_B}, P_{pub})$ and $K_A = \beta_A P - \delta_A(R_A + h_1 P_{pub}) - h_3 X_A$. ξ sets $\sigma_W = \beta_A$ and $h_2 = \delta_A$. At last ξ returns (K_A, σ_W) to Adv_2 and adds the tuple $(m_w, SK_A, RID_A, PK_{ID_A}, \delta_A)$ to list L_2. Note that delegation (K_A, σ_W) generated in this way satisfies the equation $\sigma_W P = K_A + h_2 R_{ID_A} + h_3 X_A + h_1 P_{pub}$.
- ***Queries on Proxy Key Generation:*** Upon receiving such query on (m_w, ID_A, ID_B), ξ gets (K_A, σ_W) through Delegation generation queries. If $ID_B = ID^*$, ξ stops simulation. Otherwise, ξ finds the tuples $(ID_B, x_B, PK_{ID_B}), (ID_B, d_B, R_B)$ from L_{Cuser}, L_{PSk} respectively and recovers x_B, d_B. Also recovers $(m_w, \sigma_w, ID_A, ID_B, PK_B, l_{2i}), (ID_B, V_B, t_B, l_{3i})$ from L_2', L_3' and computes $IPSK_B = d_B + x_B\, l_{2i}' + hsk_B\, l_{3i}'$ by choosing $hsk_B \in Z_q^*$ randomly, ξ returns the proxy key $IPSK_B$ to Adv_2.
- ***Queries on Temporary Signing key Oracle*** $(TSK(ID_i))$: When Adv_2 asks a query $TSK(ID_i)$ for the period t_i, ξ searches the list L_{TSK} and returns TSK_{ID_i, t_i}, if the query has already issued. Otherwise ξ performs the following.

(i) If $ID_i = ID^*$, ξ stops the simulation.

(ii) If $ID_i \neq ID^*$, ξ selects a random $v_i \in Z_q^*$ and sets $V_i = v_i P$, $IPSK_{ID_i,t_i} = d_i + x_i h'_{2i} + v_i h'_{3i}$, where $h'_{2i} = H'_2(m_w, \sigma_w, ID_A, ID_B, PK_B)$, $h'_{3i} = H'_3(ID_i, V_i, t_i)$ and sets $TSK_{ID_i,t_i} = [IPSK_{ID_i,t_i}, V_i]$. ξ outputs TSK_{ID_i,t_i} as the temporary signing key and returns it to Adv_2.

- ***Proxy Signing Oracle:*** When Adv_2 asks this query on (m, m_w, ID_A, ID_B), ξ first asks $H_1, H_2, H_3, H'_2, H'_3, H_4$ and H_5 for $i = A, B$ queries and recovers the corresponding tuple from $L_1, L_2, L_3, L'_2, L'_3, L_4, L_5$ and (ID_i, x_i, PK_i) from L_{Cuser}.

(i) If $ID_i \neq ID^*$, ξ proceeds as in the scheme.

(ii) If $ID_i = ID^*$, ξ selects a random $u_B, v_B \in Z_q^*$ and sets $V_B = v_B P$ and computes $\sigma_B = h_{4i} d_i + h_{5i} u_B$, $U_B = h_{4i}\left[u_B P - [h'_{2i} X_i + h'_{3i} V_i] h_{4i} (h_{5i})^{-1}\right]$. Finally, the challenger ξ outputs a valid Proxy signature as $\Omega_B = (R_{ID_A}, V_B, U_B, \sigma_B, ID_A)$ .Note that (m, m_w, Ω_B) in this way satisfies the verification equation.

- ***Forgery/Output:*** Finally, Adv_2 outputs the forgery $(t_i^*, m^*, \mathrm{m}_\omega^*, \Omega_B^*)$, $\Omega_B^* = (R_{ID_A}^*, V_B^*, U_B^*, \sigma^*, ID_A^*)$. If $ID_i \neq ID^*$, ξ aborts the simulation. Else ξ proceeds as follows.

Let $\Omega_B^{*(1)} = \left(R_{ID_A}^*, V_B^*, U_B^*, \sigma^{*(1)}, ID_A^*\right)$, when we repeat using the same random tape but a different selection of hash functions $H_1, H_2, H_3, H'_2, H'_3, H_4, H_5$, according to Forking Lemma [29]. Adv_2 will output another four signatures $\Omega_B^{*(j)} = \left(R_{ID_A}^*, V_B^*, U_B^*, \sigma^{*(j)}, ID_A^*\right)$ for $j = 1,2,3,4$, and the following equation holds. $\sigma^{*(j)} P - h_5^{*(j)} U_B^* = h_4^{*(j)}\left(R_{ID_A}^* + h_1^{*(j)} P_{pub} + h_2^{*(j)} X_B + h_3^{*(j)} V_B^*\right)$ for $j = 1,2,3,4.$ (3)

By u_B, r_i, α, v_B we denote discrete logarithms of U_B, R_{ID_A}, X_B and V_B respectively. i.e. $R_{ID_A}^* = r_i P$, $U_B^* = u_B P$, $X_B^* = \alpha P$ and $V_B = v_B P$. From equation (3) we get,

$$\sigma^{*(j)} - h_5^{*(j)} u_B^* = h_4^{*(j)} \left(r_i^* + h_1^{*(j)} s + h_2^{*(j)} \alpha + h_3^{*(j)} v_B^* \right) \text{for } j = 1,2,3,4. \qquad (4)$$

ξ solves the unknowns r_i^*, u_B^*, α and v_B^* by solving these linearly independent equations (4) and outputs 'α' as the solution of ECDLP instance.

Theorem 3: According to the ECDLP assumption and in RO model, our PF-CLKIPS scheme satisfies the strong key insulated property against Type-I adversary Adv_1.

Proof: The proof is similar to that of Theorem 1. But, Adv_1 never ask a temporary signing key query.

Helper Key Query: When Adv_1 asks a helper key for ID_i for a time period t_i, ξ executes the helper key extraction algorithm and returns $v_i \in Z_q^*$ to Adv_1.

Signing Oracle: When Adv_1 makes a query on (t_i, m_i, ID_i). ξ does the following.

(i) If $ID_i \neq ID^*$, ξ recovers v_B and compute $V_B = v_B P$, $\sigma_B = h_{4i}\left[h'_{3i} v_B + h'_{2i} x_i\right]$ and $U_B = h_{4i}\left[R_i + h_{1i} P_{pub}\right](-h_{5i})^{-1}$, ξ outputs $\Omega_B = \left(R_{ID_A}, V_B, U_B, \sigma_B, ID_A\right)$ as the signature.

(ii) If $ID_i = ID^*$, ξ recovers v_i for D^* and performs the same as in Theorem1.

Theorem 4: Under the ECDLP assumption and in RO model, our PF-CLKIPS Scheme satisfies the strong key insulated property against Type-II adversary.

Proof: The proof is similar to Theorem 2. However, Adv_2 never asks a temporary signing key query and he can make an Helper key query as in Theorem 3.

Theorem 5: Against adversaries of Types I and II, our PF-CLKIPS scheme has secure key updates.

Proof: The proof follows from the fact that for any time period t_i, t_{i-1} and any identity ID_i, the updated secret key $UHK_{ID_i, t_i, t_{i-1}}$ can be derived from TSK_{ID_i, t_i} and $TSK_{ID_i, t_{i-1}}$.

6. PERFORMANCE COMPARISON

We take into consideration a few cryptographic operations and their execution times, which are listed in Table-2, in order to assess the effectiveness of our CL-KIPS system. We take into consideration the experimental results from [9,30,31], where system is based on bilinear pairings $\hat{e}: G_1 \times G_1 \to G_2$ in order to attain the similar security with a 1024-bit RSA key. Here G_1 denotes a group with $q-$order and generator P. $\hat{E}: y^2 = x^3 + x (\bmod \hat{p})$, where $\hat{p}$ represents a prime number of 512 bits and q represents a prime number of 160 bits. ECC-based proposals employ an additive $q-$order group G with its generator P on elliptic curves $E: y^2 = x^3 + ax + b (\bmod p)$, , where $a, b \in Z_q^*$, p and q are both prime numbers of 160 bits. Running times are calculated using MIRACL cryptographic library [32] and implemented on the hardware platform P-IV (Pentium-4) 3GHZ processor with 512-MB memory and a windows XP operating system.

6.1. Computational Cost

We now analyze the computation cost of our CL-KIPS scheme and then we compare it with the existing Wan et al. [8] KIPS scheme.

Table 2: Different cryptographic operations and their execution times

Notations	***Description***
$1T_{MM} \approx 0.2325ms$	Modular multiplication operation
$T_{SM} \approx 6.38ms$	Pairing based Scalar multiplication
$T_{BP} \approx 20.01ms$	Bilinear pairing
$1T_{MPH} \approx 6.38ms$	Map to point hash function
$T_{PA} \approx 0.0279ms$	Pairing based point addition
$T_{SM-ECC} \approx 0.83ms$	Scalar multiplication on EC
$T_{PA-ECC} \approx 0.0034ms$	Point addition on EC

Table 3: Comparison of Computation cost

Scheme	**Proxy Signature Generation Cost**	**Proxy Signature Verification Cost**	**Total Computation Cost (in ms)**
Wan et al. [8]	$3T_{SM}+1T_{PA}$ $=3(6.38)+0.0279$ $=19.1679\ ms$	$4T_{SM}+4T_{BP}+$ $2T_{PA}+2T_{MPH}$ $=118.3758\ ms$	137.5437
Proposed CL-KIPS Scheme	$3T_{MM}+$ $1T_{SM-ECC}$ $=1.5275\ ms$	$6T_{SM-ECC}+$ $4T_{PA-ECC}$ $=4.9936\ ms$	6.5211

To calculate the computation cost, we consider signing, verification and the total costs. For proxy signature generation, Wan et al. [8] scheme requires 3 scalar multiplications and one point addition. i.e. Wan et al. [8] scheme needs $3T_{SM}+1T_{PA}=3(6.38)+0.0279=19.1679\ ms$ for proxy signature generation. For verification, Wan et al. [8] scheme requires 4 scalar multiplications, 4 bilinear pairing operations, 2 point additions and 2 map to point hash function evaluations i.e. it requires 118.375 ms for proxy signature verification. Thus totally, Wan et al. [8] scheme needs 137.5437 *ms*. Similarly, for proxy signature generation, the proposed CL-KIPS scheme requires 3 modular multiplications and 1 scalar multiplication i.e, our scheme requires $3T_{MM}+1T_{SM-ECC}=1.5275\ ms$ for proxy signature generation. Also, for signature verification it requires $6T_{SM-ECC}+4T_{PA-ECC}=4.9936\ ms.$ Thus totally the proposed scheme requires 6.5211*ms*. These computation costs are presented in Table 3. From the Table 3, our scheme improves the computational efficiency by $\left(\frac{137.5437-6.5211}{137.5437}\right)X100=95.25\%$ over the Wan et al. [8] scheme. it is clear that the proposed PF-CLKIPS scheme is computationally more efficient than Wan et al. proxy key insulated signature scheme. The comparison of computational cost our scheme with Wan et al. [8] scheme is presented in Figure 1.

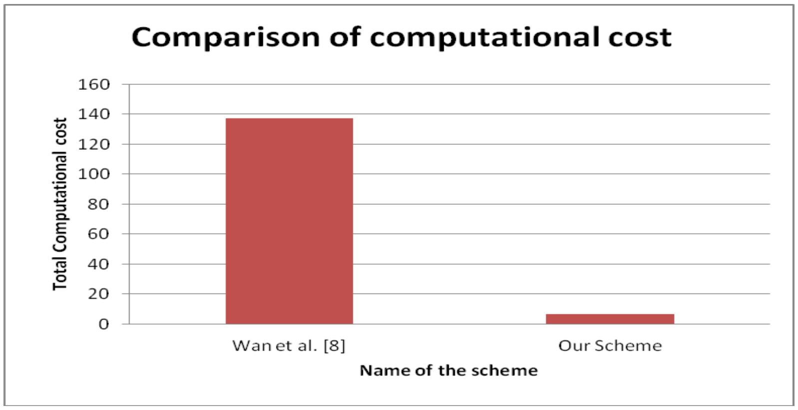


Fig. 1. Comparison of Computation cost

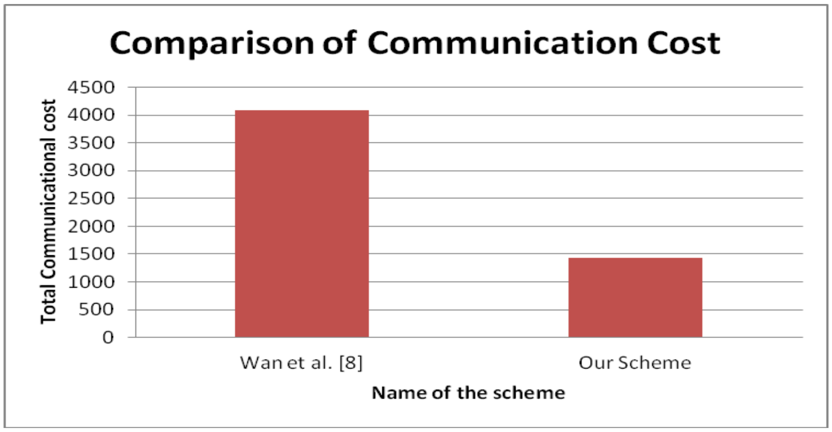
6.2. Communication Cost

In order to asses the communication cost, the signature length was taken into account. Since the Wan et al. [8] scheme is pairing based scheme and it the signature size of the Wan et al. scheme is $4|G_1| = 4(1024) = 4096$ bits. The proposed CL-KIPS scheme is constructed in pairing-free environment and the signature size of our scheme is $4|G| + |Z_q^*| = 4(320) + 160 = 1440$ bits.

Table 4: Comparison of Communication cost

Scheme	***Signature Length***	***Communication Cost***
Wan et al. [8]	$4\|G_1\|$	4096 bits
Our CL-KIPS Scheme	$4\|G\| + \|Z_q^*\|$	1440bits

Table 4 presents a comparison of different schemes' communication costs. From Table 4, we can observe that our scheme improves the communication efficiency by $\left(\frac{4096-1440}{4096}\right) X100 = 64.84\%$ over the Wan et al. [8] scheme. For this reason, the suggested CL-KIPS method is effective from a communication point of view. The comparison of communication cost our scheme with Wan et al. [8] scheme is presented in Figure 2.



.

Fig. 2. Comparison of Computation cost

According to the above discussion, the proposed CL-KIPS scheme is efficient in computational and communication point of view than the existing Wan et al. [8] proxy key insulated signature scheme.

7. CONCLUSION

This paper presents a novel and effective key-insulated proxy signing system that does not require bilinear pairings over elliptic curves and is based on a certificateless framework. A proxy signer can sign a message on behalf of the original signer using this scheme. In spite of the loss of the proxy signing key, the suggested system remains secure because to the key insulation technique. Assuming the difficulty of the ECDL problem, the security analysis of the suggested technique demonstrates that it is proven secure and unforgeable. Compared to current systems, the proposed scheme has a lower computational and communication overhead because of its pairing-free environment. According to the efficiency analysis, our scheme outperforms the current Wan et al key insulated proxy signature technique in terms of computing efficiency by 95.25% and communication efficiency by 64.84%. Therefore, the suggested scheme is a good fit for implementation on resource-constrained devices, such as wireless sensor networks (WSNs), personal digital assistants (PPAs), mobile phones, radio frequency identification (RFID) chips, and sensor devices, which have limited processing power, storage capacity, and communication bandwidth.

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests.

REFERENCES

[1] W. Diffie, M. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory, 22 (1976), 644-654.

[2] A. Shamir, Identity-based cryptosystems and signature schemes, crypto '84, Lecture Notes in Computer Science, Springer-Verlag, 196, 47-53, (1985).

[3] S.S. Al-Riyami, K.G. Paterson, Certificateless public key cryptography, Lecture Notes in Computer Science, Springer-Verlag, 2894, 452-473, (2003).

[4] M. Mambo, K. Usuda, E. Okamoto, Proxy signatures for delegating signing operation, in: 3rd ACM Conf. on Computer and Communications Security (CCS'96), New York: ACM Press, 48-57, (1996).

[5] Y. Dodis, J.J. Katz, S. Xu, M. Yung, Key-insulated public key cryptosystems, in: Proc. of Int. Con. on the Theory and Apps of Crypto. Tech's, Amsterdam, 65−82, (2002).

[6] Y. Dodis, J.J. Katz, S. Xu, M. Yung, Strong key-insulated signature schemes, in: Proc. of the 6th Int. Workshop on Practice and Theory in PKC, Miami,130−144, (2002).

[7] J. Wang, Q. Wu, Y. Wang, A new perfect and strong key insulated signature scheme, Proc. of China Crypto'04, 233-239, (2004).

[8] Z. Wan, X. Lai, J. Weng, Y. Wang, Identity-based key-insulated proxy signature, J. Electron. 26 (2009), 853−858.

[9] G. Thumbur, G.S. Rao, P.V. Reddy, N.B. Gayathri, D.V.R.K. Reddy, Efficient pairing-free certificateless signature scheme for secure communication in resource-constrained devices, in: IEEE Comm. Letters, 24, 1641-1645, (2020).

[10] V. Miller, Uses of elliptic curves in cryptography, in: Proceedings of Advances in Cryptology-Crypto. 85, 417-426, (1985).

[11] N. Koblitz, Elliptic curve cryptosystems, Math. Comput. 48 (1987), 203-209.

[12] C.A. Lara-Nino, A. Diaz-Perez, M. Morales-Sandoval, Elliptic curve lightweight cryptography: a survey, IEEE Access, 6 (2018), 72514-72550.

[13] S. Di Matteo, L. Baldanzi, L. Crocetti, et al. Secure elliptic curve crypto-processor for real-time IoT applications, Energies. 14 (2021), 4676.

[14] A.R. Babu, N.B. Gayathri, P.V. Reddy, Efficient ID-based key-insulated multi signature scheme without pairings, in: proc. of Innovations in Power and Advanced Computing Technologies (i-PACT), 1-6, (2019).

[15] O.M. Ebadti, F. Eshghi, A. Zamni, Security enhancement of wireless sensor networks, J. Inform. Syst. Telecommun. 6 (2018), 177-188.

[16] S. Iqbal, B.R. Sujatha, Secure key management scheme for hierarchical network using combinatorial design, J. Inform. Syst. Telecommun. 10 (2021), 20-27.

[17] H. Farsi, S.M. Nourian, Node to node watermarking in wireless sensor networks for authentication of self nodes, J. Inform. Syst. Telecommun. 2 (2014), 127-136.

[18] A.R. Babu, N.B. Gayatri, Efficient and Secure identity based Strong Key-insulated signature scheme without Pairings, J. King Saud Univ. – Computer Inform. Sci. 33 (2021), 1211-1218.

[19] V.S.S.N. Gopal, P.V. Reddy, Efficient ID-based key-insulated signature scheme with batch verifications using bilinear pairings over elliptic curves, J. Discr. Math. Sci. Cryptography, 18 (2015), 385-402.

[20] X. Hong, K. Chen, Secure key-insulated proxy signature scheme for mobile agent, in: IEEE Proc. of the Second International Conference on Innovative Computing, Information and Control (ICICIC), 513-518, (2007).

[21] L.B. He, D.J. Yan, H. Xiong, et al. A Pairing-free Certificateless key insulated encryption with provable security, J. Electron. Sci. Technol. 16 (2018), 1-7.

[22] A.R. Babu, N.B. Gayathri, P.V. Reddy, Efficient and secure pairing-free certificateless strong key-insulated signature scheme, Int. J. Knowl.-based Intell. Eng. Syst. 23 (2019), 155-166.

[23] J. Weng, X. Li, K. Chen, et al. Identity-based parallel key-insulated signature without random oracles, J. Inform. Sci. Eng. 24 (2008), 1143−1157.

[24] C. Zhou, Z. Zhao, W. Zhou, et al. Certificateless key-insulated generalized signcryption scheme without bilinear pairings, Secur. Commun. Networks. 2017 (2017), 8405879.

[25] J. Chen, Y. Long, K. Chen, et al. Identity-based key-insulated proxy signature without random oracles, J. Shanghai Jiaotong Univ. (Sci.). 16 (2011), 557–566.

[26] J. Chen, Identity based threshold key-insulated proxy signature in the standard model, J. Phys.: Conf. Ser., 1345 (2019), 1-5.

[27] J. Chen. Identity based parallel key-insulated proxy signature in the random oracle model, in: International Conference on E-Commerce and Internet Technology (ECIT), 277-280, (2020).

[28] H. Xiong, Z. Qin, A.V. Vasilakos, Introduction to certificateless cryptography, CRC Press, 2015.

[29] D. Pointcheval, J. Stern, Security arguments f or digital signatures and blind signatures, J. Cryptol. 13 (2000), 361-369.

[30] P. Barreto, Y.H. Kim, B. Lynn, et al. Efficient algorithms for pairing based cryptosystems, Lecture Notes in Computer Science 2442, 354-368, (2002).

[31] X. Cao, W. Kou, X. Du. A pairing–free identity based authenticated key agreement protocol with minimal message exchanges, Inform. Sci. 180 (2010), 2895-2903.

[32] MIRACL Library, http://certivox.org/display/EXT/MIRACL.