Information Sciences 450 (2018) 378–391

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A pairing-free and provably secure certificateless signature scheme

Arijit Karati [a], SK Hafizul Islam [b,*], G.P. Biswas [c]

[a] Department of Computer Science and Engineering, NIIT University, Neemrana, Rajasthan 301705, India
[b] Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India
[c] Department of Computer Science and Engineering, Indian Institute of Technology (ISM), Dhanbad, Jharkhand 826004, India

ARTICLE INFO

Article history:
Received 11 October 2017
Revised 19 March 2018
Accepted 22 March 2018
Available online 27 March 2018

Keywords:
Digital signature
Elliptic curve
Certificateless cryptography
Random oracle model
Provable security

ABSTRACT

Certificateless Signature (CLS) scheme is a notable cryptographic technique for solving the key escrow problem in identity-based cryptosystem (IBC). In the CLS, the private key is computed collectively by both the key generation center (KGC) and the signer which ensures that no vindictive KGC masquerades the actual signer. Recently, a number of CLS schemes have been proposed using bilinear pairing and show their immunity under standard security model. It is well known that one such pairing operation requires significantly more computational cost than the other cryptographic operations. In this paper, we propose a new CLS scheme using elliptic curve cryptography (ECC), which does not require bilinear pairing operation. Our CLS scheme is analyzed formally and found to be provably secure against both the Type-I and Type-II attacks based on the intractability of elliptic curve discrete logarithm problem (ECDLP) under the random oracle model. Performance evaluation demonstrates that the proposed CLS scheme outperforms than other competitive CLS schemes.

1. Introduction

Traditional public key cryptography (PKC) needs additional support which is known as public key infrastructure (PKI). The authenticity of users' public key is an essential aspect and therefore, it is necessary to be satisfied prior to the secure communication. Hence, to achieve the key authenticity, PKI maintains certificates by ensuring that the user's public key is not tampered by any malicious entity. A trusted entity called certificate authority (CA) is assumed to issue and distribute the certificates by binding users' identity with their public keys. However, PKI is considered as a high computational infrastructure in the real-life scenarios due to the certificate management overhead (e.g., storage, distribution, verification, and revocation). A novel concept of IBC was proposed by Shamir [23] in 1984 to solve the aforementioned problem. The motive of that proposal was to choose a unique identity (e.g., email-id, phone-no, IP-address, etc.) of each party as their public key without being certified by the trusted authority. Besides, a trusted entity called key generation center (KGC) is considered to compute and send the participant's secret key over the secure channel. Since the KGC computes users' private key, it has all the potentials that a party can perform during the signature generation. Therefore, the KGC could act as a target entity for any attacker. This is commonly known as key-escrow problem. To resolve the aforementioned deficiency in IBC, a new

* Corresponding author.
E-mail addresses: arijit.karati@gmail.com (A. Karati), hafi786@gmail.com (S. Hafizul Islam), gpbiswas@gmail.com (G.P. Biswas).

https://doi.org/10.1016/j.ins.2018.03.053
0020-0255/

cryptosystem known as certificateless public key cryptography (CL-PKC) was devised by Al-Riyami and Paterson [1]. This cryptosystem inherits the benefits of PKC as well as IBC. The CL-PKC introduced a unique idea of computing the key pair (public and private keys) by utilizing both the user's and KGC's private keys. The CL-PKC allows every user to have a private key, which is a pair of keys. One component of the pair is called secret value whereas another component is known as the partial-private-key. The secret value is selected at random by the user itself while the partial-private-key is computed by the KGC using its master secret key. In order to authenticate a message, the signature generation algorithm requires both the keys (i.e., secret value and partial-private-key) of the user. Thus, the key escrow problem is eliminated by disabling the full access to parties' private key to KGC. In the CL-PKC, users' public key, which would be known to everyone, is also a pair of keys. One component of the pair is determined by the user itself using its own secret value, and the other component is calculated by the KGC using the partial-private-key. In this settings, users' public key is to be marked obtainable to others by sending it together with the signature or by making it achievable in any publicly available directory in an appropriate manner.

1.1. Literature survey

In 2013, Al-Riyami and Paterson [1] introduced a novel certificateless signature (CLS) scheme, which eliminates the key-escrow problem. This pioneering work initiated a paradigm shift in the PKC technique. Since Al-Riyami and Paterson's work, many certificateless encryption [6,25,26] and signature [4,8,13,33,37] schemes developed in recent years. In 2005, Gorantla and Saxena [10] designed an lightweight CLS, but unfortunately, Cao et al. [3] illustrated that the scheme in [10] can not resists Type-I attack. Informally, Type-I adversary $\mathcal{A}_I$ is a third party who can mesh user's public keys corresponding to the user's private value. In 2006, Yap et al. [34] devised a pairing-based CLS technique under the intractability of computational Diffie–Hellman (CDH) problem in random oracle model (ROM). Their scheme was computationally efficient as the signing phase was pairing-free and the verification phase needs only two pairing computation. But, Zhang and Feng [36] analyzed and showed that the scheme in [34] is attackable under the public key replacement attack. Shortly thereafter, Huang et al. [14] devised a pairing-based CLS scheme, which produces considerably short-size signature. However, Shim [24] demonstrated that the scheme in [14] cannot withstand Type-I attack. After that, Tso et al. [32] devised a CLS scheme, but it is also found to be insecure against Type-I attack. Du and Wen [7] designed a novel CLS technique in 2009, which produces a short-size signature and does not use map-to-point (MTP) hash function during the construction. However, their scheme is proven to be insecure against Type-I attack. Recently, Choi et al. [5] devised a novel CLS with a short-signature facility, although, Tian et al. [28] proved that the scheme is still suffering from Type-I adversary. Like, Choi et al.'s scheme [5], the CLS proposed by Tso et al. [31] was found to be unsecured against Type-I adversary. Since the execution cost of the bilinear pairing is very high as compared to the other cryptographic operations, He et al. [12], presented a CLS scheme in 2012 without using bilinear pairing operation. It was a good attempt to construct their CLS for the bandwidth concerned communication, although, Tian and Huang [27] showed that the scheme in [12] did not withstand for Type-II adversary $\mathcal{A}_{II}$. Informally, Type-II adversary masquerades the KGC, which could have the access to all the partial-public(private)-key of all the users. Recently, Tsai [29] designed a CLS by utilizing the bilinear pairing and ECC operations. It was computational efficient and secured under the collusion attack algorithm with k-traitors (k-CAA) in the formal security structure, but it was shown as improper digital signature by Karati and Biswas [18].

1.2. Motivations and contributions

Many of the recently proposed CLS schemes have been designed based on either of bilinear pairing or map-to-point (MTP) hash function. Besides, it is observed that the schemes are insecure against either Type-I attack or Type-II attack. Although, the bilinear pairing and MTP hash function are frequently used to implement any cryptographic scheme easily, the computational cost of such operations are very high as compared to the other cryptographic operations [15–17]. Therefore, the security and performance deficiencies (discussed in Section 1.1) encourage us to built a lightweight and secure CLS scheme using ECC. The proposed CLS scheme has the following characteristics:

- It works on the super-singular elliptic curve points. Besides, it produces short-length signature by considering the short-length public parameters.
- The security analysis depends on the hardness assumption of ECDLP in the ROM [2]. It is existentially unforgeable against both Type-I and Type-II adversaries.
- It does not use any pairing and MTP hash operations during the signature generation and verification phases. Therefore, the proposed CLS is computationally efficient than other competitive CLSs as those schemes utilize either of these two operations.
- Since the proposed CLS scheme has short-length public parameters, short-length signature, existentially unforgeability and computational efficiency, it can be mounted in low-power and low-storage devices like mobile phone, PDA, smart-card, etc.

Table 1
List of the symbols used in our CLS scheme.

Notation	Meaning
q	Prime number, considerably large (512-bit) in size
G_q	Additive cyclic group of elliptic curve points of order q
P	Generator of the group G_q
$H(\cdot)$, $H_1(\cdot)$, $H_2(\cdot)$	One-way hash functions, defined as: $H : \{0,1\}^* \times G_q \times G_q \rightarrow Z_q^*$, $H_1 : \{0,1\}^* \times G_q \times G_q \rightarrow Z_q^*$, and $H_2 : \{0,1\}^* \times G_q \times G_q \times G_q \rightarrow Z_q^*$
MSK	Master secret key of the KGC
$params$	Public key of the KGC
ID_i	Identity of the user i
x_i	Secret value of the user i
D_i	The partial-private-key of the user i
PK_i	The public key of the user i
m	Message $m \in \{0, 1\}^*$
σ	Signature on m
$A \stackrel{?}{=} B$	Check whether A is equal to B or not
$x \in_R A$	x is selected at random from the set A

1.3. Outline of the paper

The remaining of the article is structured in the following manner. Section 2 demonstrates some preliminaries on CLS, ECC, and mathematical hard assumptions. The main construction of the proposed CLS with its correctness are mentioned in Section 3. The security and efficiency analysis of our CLS scheme are discussed in Section 4. Finally, Section 5 draws some concluding remarks.

2. Preliminaries : Concepts and definitions

This section discusses the formal definition and security model of CLS, basics of the ECC and some hardness assumptions. Before that, various symbols used in the article are described in Table 1.

2.1. Formal definition of the CLS

The scheme considers six algorithms namely *Setup, Set-Partial-Private-Key, Set-Secret-Value, Set-Public-Key, CLS-Sign*, and *CLS-Verify*, which are explained briefly.

- Setup: For the input parameter k, the KGC computes a master secret key MSK, and the global parameters $params$.
- Set-Partial-Private-Key: On receiving the parameters $params$, MSK, and the signer's identity ID_S, the KGC outputs a Partial-Private-Key D_S to the signatory S. After that, the signatory S can check the validity of the received D_S whenever it is required.
- Set-Secret-Value: Signatory S executes this algorithm and returns the Secret-Value x_S.
- Set-Public-Key: On input x_S as the required parameter, S executes the algorithm to computes its Public-Key PK_S.
- CLS-Sign: The signatory S generates a signature σ for the inputs $params$, m and D_S. After that, it sends σ to the verifier V.
- CLS-Verify: For the required input parameters $params$, PK_S, m and σ, the verifier receives VALID if σ is correct, otherwise, receives INVALID.

2.2. Security model of CLS scheme

There are two type of security breaches called Type-I and Type-II attacks found in the CLS scheme [1]. Now, we discuss these two attacks in terms of **Game 1** and **Game 2**, respectively.

Game 1: This is basically a challenge-response game. Here, the main objective of the adversary $\mathcal{A}_I$ is to breach the semantic security of CLS by playing with the challenger $\mathcal{C}$. For this purpose, $\mathcal{A}_I$ and $\mathcal{C}$ interactively execute this game, i.e., $\mathcal{A}_I$ submits some queries to $\mathcal{C}$, and $\mathcal{C}$ responds to $\mathcal{A}_I$ with some appropriate answers. Finally, $\mathcal{A}_I$ will breach the security of the CLS based on the answers received from $\mathcal{C}$. The brief overview of this game is given below.

- Setup: Initially, the challenger $\mathcal{C}$ computes KGC's private key s, and $params$. After that, it keeps s securely while marks $params$ public to the adversary $\mathcal{A}_I$.
- Queries: The $\mathcal{A}_1$ imposes one of the following queries adaptively to $\mathcal{C}$:
  - Set-Partial-Private-Key (ID_i): $\mathcal{A}_I$ learns about the Partial-Private-Key D_i of the user i.
  - Set-Secret-Value (ID_i): $\mathcal{A}_I$ obtains a returned Secret-Value x_i of the user i.
  - Set-Public-Key (ID_i): $\mathcal{A}_I$ gets the Public-Key PK_i of the user i.
  - Replace-Public-Key (ID_i, PK_i): $\mathcal{A}_I$ is capable to change user i's recent PK_i with a fresh random Public-Key PK^*.

  - CL-Sign (ID_i, m): $\mathcal{A}_I$ obtains the signature σ for which Verify($params$, ID_i, PK_i, m, σ) is VALID, where PK_i is the most recently considered public key of user i.
- Output: $\mathcal{A}_I$ sends a fake signature σ' for a selected message m' with the targeted signatory's identity ID', and wins the game if the two essential conditions are fulfilled.
  - Set-Partial-Private-Key and CL-Sign queries have not been asked so far.
  - Verify($params$, m', σ, ID', PK')=VALID though $\mathcal{A}_I$ replaces Public-Key PK' of the user with identity ID'.

Definition 1. A CLS scheme is $(t, q_{PR}, q_{PU}, q_{RPU}, q_S, \epsilon)-$ Type-I secure against the adaptively chosen-message and -ID attack, if any $\mathcal{A}_I$ gains a negligible advantage ϵ for the maximum q_{PR} number of Set-Partial-Private-Key, q_{PU} number of Set-Public-Key, q_{RPU} number of Replace-Public-Key and q_S number of CL-Sign queries in the polynomial time t.

Now, **Game 2** is similar as like in **Game 1**, except that the adversary $\mathcal{A}_{II}$ in **Game 2**, has the knowledge of KGC's private key, i.e., $\mathcal{A}_{II}$ is able to compute the Partial-Private-Key of any user. This is known as Type-II forger. The game is defined as follows:

Game 2: The main objective of the adversary $\mathcal{A}_{II}$ to play this game with a challenger $\mathcal{C}$ to breach the semantic security of the CLS scheme. For this purpose, $\mathcal{A}_{II}$ and $\mathcal{C}$ interactively execute this game.

- Setup: $\mathcal{C}$ produces KGC's private key s and global parameters $params$. It makes s, $params$ public to the $\mathcal{A}_{II}$.
- Queries: $\mathcal{A}_{II}$ imposes one of the following queries at a time to $\mathcal{C}$:
  - Set-Partial-Private-Key (ID_i): $\mathcal{A}_{II}$ learns about the Partial-Private-key D_i of the user i.
  - Set-Secret-Value (ID_i): $\mathcal{A}_{II}$ obtains a returned Secret-Value x_i of the user i.
  - Set-Public-Key (ID_i): $\mathcal{A}_{II}$ gets the Public-Key PK_i of the user i.
  - CL-Sign (ID_i, m): $\mathcal{A}_{II}$ obtains signature σ such that the output of Verify($params, ID_i, PK_i, m, \sigma$) is VALID, where PK_i is most recently used Public-Key of user i.
- Output: Finally, $\mathcal{A}_{II}$ sends a fake signature σ' for a selected message m' with the targeted signatory's ID', and wins this game if the two following essential conditions are fulfilled.
  - Set-Secret-Value and CL-Sign queries have not been asked so far.
  - Output of Verify($params$, m', σ, ID', PK') is VALID.

Definition 2. A CLS scheme is $(t, q_{PR}, q_{PU}, q_S, \epsilon)-$ Type-II secure against the adaptively chosen-message and -ID attack, if $\mathcal{A}_{II}$ gains a negligible advantage ϵ for maximum q_{PR} number of Partial-Private-Key, q_{PU} number of Partial-Public-Key and q_S number of CL-Sig queries in the polynomial time t.

Definition 3. A CLS scheme is secure under the adaptively chosen-message and -ID attack, if it is invulnerable against $\mathcal{A}_I$ and $\mathcal{A}_{II}$ adversaries.

2.3. Elliptic curve cryptography (ECC)

The ECC works on the theory of elliptic curve. It was initially introduced by Koblitz [19] and Miller [20] in 1985, and received wide popularity from 2004. The security of ECC is based on the intractability assumption of ECDLP. Any scheme under ECC provides high security with smaller size key, e.g., a 160-bit ECC key provides the same security as a 1024-bit RSA key [11] does. It is used to design different cryptographic protocols/schemes for resource-constrained devices, namely smartcard, PDA, smart device, etc. Recently, the ECC gets significant attention in the field of key agreement, digital signature, encryption, user authentication which are useful in the area of Internet-of-things (IoT), cloud computing, vehicular ad hoc network (VANET), wireless networks, etc.

Let, F_q be the integer field of prime modulo q. The condition for a non-singular elliptic curve $E_q(a, b)$ on F_q is formulated as $y^2 \equiv (x^3 + ax + b) \pmod q$ where $a, b, x, y \in F_q$ and $\delta = (4a^3 + 27b^2) \bmod q \neq 0$. Any point $S(x, y)$ is said to be an elliptic curve point if it follows the above mentioned condition. A point $T(x_1, y_1)$ is termed as the negative of S, i.e., $T = -S$ where $x_1 = x$ and $y_1 = -y$. If $S(x_1, y_1)$ and $T(x_2, y_2)$ are the two different points and satisfy the required condition, then line l (tangent line if $S = T$) joining the points S and T intersects the curve at $R(x_3, y_3)$, and the reflection of R under $x-$axis is point $R(x_3, y_3)$, i.e., $S + T = R$. The points $E_q(a, b)$ with the O (point at infinity) define a prime ordered additive cyclic group $G_q = \{(x, y) \mid a, b, x, y \in F_q \text{ and } (x, y) \in E_q(a, b)\} \cup \{O\}$. The scalar point multiplication on group G_q is computed as $kS = S + S + \cdots + S$ (k times). Here, the order n of any point S indicates that n is the least positive integer such that $nS = O$.

2.4. Mathematical definitions

This section discusses some useful hard assumptions where the solutions are infeasible for all the probabilistic polynomial-time (PPT) bounded algorithms $\mathcal{B}$.

Definition 4 (One-way cryptographic hash function)**.** For an algorithm $\mathcal{B}$, the computation of y from a conferred $H(y)$ is infeasible. The advantage ϵ of $\mathcal{B}$ in resolving another solution rather than y is formulated as

$$\Pr\left[y \leftarrow_R \{0,1\}^*;\ z \leftarrow H(y);\ y' \leftarrow \mathcal{B}(z):\quad H(y') = z\ \right] \geq \epsilon$$

Definition 5 (ECDLP assumption)**.** Given a tuple $\langle P, Q\rangle$ for some $P,\ Q \in G_q$, it is computationally infeasible for every $\mathcal{B}$ to identify $x \in Z_q^*$ so that $Q = xP$. The advantage ϵ of $\mathcal{B}$ in resolving the solution of the ECDLP is formulated as

$$\Pr\left[P, Q \in G_q;\ x \leftarrow \mathcal{B}(P, Q):\quad Q = xP\ \right] \geq \epsilon$$

Definition 6 (CDH assumption)**.** Given a tuple $\langle P, xP, yP\rangle$ for some $x, y \in_R Z_q^*$, it is computationally infeasible for every $\mathcal{B}$ to calculate xyP. The advantage ϵ of $\mathcal{B}$ in finding CDH solution is formulated as

$$\Pr\left[x, y \in_R Z_q^*;\ P \in_R G_q;\ Q \leftarrow \mathcal{B}(P, xP, yP):\quad Q = xyP\ \right] \geq \epsilon$$

2.5. Concept of forking lemma

In many signature schemes, having two different signatures generated using the single random value for the two uniformly different hash values allows an attacker to disclose the original private key. The concept was primarily introduced by Pointcheval and Stern [21] and applied rapidly in many security proofs. The technique demonstrates that if an adversary is persecuted through replaying into an unchanged random tape for different choices of a single entity, then it is possible to generate two more valid signatures from a single forged signature. As a consequence, the adversary can perform further a key-recovery attack. This strategy is used later to prove the security of the proposed CLS scheme.

Definition 7. IG (Input generator) chooses integers x, $q \geq 1$ at random, and the sets $H = \{h_i\}_{\forall i \in [1,q]}$ such that $|H| \geq 2$. For an input tuple $(x, H; \rho)$, a randomized algorithm $\mathcal{B}$ outputs a pair (Q, y), where ρ indicates the random tape of $\mathcal{B}$, $Q \in Z_q$, and y is referred as a side output. The forking procedure $F_{\mathcal{B}}$ accompanied with $\mathcal{B}$ is the randomized algorithm which considers input x, and executes in the following way:

Algorithm $F_{\mathcal{B}}(x)$
choose a tape ρ at random
$h_1, \ldots, h_q \leftarrow_R H$
$(Q, y) \leftarrow \mathcal{B}(x, h_1, \ldots, h_q; \rho)$
If (Q=0) then
Return $(0, \perp, \perp)$
$h'_Q, \ldots, h'_q \leftarrow_R H$
$(Q', y') \leftarrow \mathcal{B}(x, h_1, \ldots, h_{Q-1}, h'_Q, \ldots, h'_q; \rho)$
If ($Q = Q'$ and $h_Q \neq h'_Q$) then
Return $(1, y, y)$
Else~~ Return $(0, \perp, \perp)$

Assume, the accepting probability for $Q \geq 1$ of $\mathcal{B}$ is ϵ. If $F_{\mathcal{B}}$ returns a tuple $(1, -)$ for a random input x chosen by IG with probability ϵ_F, then

$$\epsilon \leq \frac{q}{h} + \sqrt{q \cdot \epsilon_F}, \quad \text{where}$$

$$\epsilon_F = \Pr\left[\ \gamma = 1\ :\ x \leftarrow_R IG;\ (b, y, y') \leftarrow_R F_{\mathcal{B}}(x)\ \right]$$

Suppose, a randomized algorithm $\mathcal{B}$ exists to breach a signature scheme under ROM. Besides, x is the public parameters, and h_i is the returned value of the oracle for i^{th} distinct input. In this scenario, the forking lemma can be utilized to solve any underlying hard problem for two random signatures of a single message. According to the forking lemma, the success probability (ϵ_F) of collecting two different forged values y and y' for message m with distinct oracle outputs (i.e., $h_Q \neq h'_Q$) is non-negligible once ϵ is non-negligible. Therefore, it enables us to state that if the underlying hard assumption is computationally intractable, then no real-time adversary can counterfeit signatures.

3. Proposed CLS scheme

This section explains a new CLS scheme, which is implemented without bilinear pairing and MTP hash operations.

3.1. Concrete construction of the CLS scheme

The six different algorithms used in the proposed CLS are discussed below.

- Setup: On giving the security parameter k as input, the KGC generates a elliptic curve group G_q of prime order q as discussed in Section 2.3. After that, it selects a generator $P(\neq O)$ of G_q, an integer $l \in_R Z_q^*$. Besides, the KGC considers three one-way hash functions as $H : \{0,1\}^* \times G_q \times G_q \rightarrow Z_q^*$, $H_1 : \{0,1\}^* \times G_q \times G_q \rightarrow Z_q^*$ and $H_2 : \{0,1\}^* \times G_q \times G_q \times G_q \rightarrow Z_q^*$. After that, it calculates $P_{pub} = \{P_{pub1}, P_{pub2}\} = \{lP, l^{-1}P\}$. Finally, the KGC keeps $MSK = l$ safely, and publishes the public parameter as $params = \{G_q, q, P, P_{pub}, H, H_1, H_2\}$.
- Set-Partial-Private-Key: On receiving *params, MSK*, and the signatory's identity ID_S as input, the KGC computes

$$R_S = rP \tag{1}$$

$$h = H(ID_S, R_S, P) \tag{2}$$

$$y = \left[l^{-1} + \left(\frac{l}{r}\right) h \right] \bmod q \tag{3}$$

$$Q_S = rl^{-1}P \tag{4}$$

where $r \in_R Z_q^*$. After that, it sends a partial-private-key $D_S = (y, R_S, Q_S)$ to the signer S. On receiving D_S from the KGC, the signer S can check the authenticity of D_S by satisfying Eq. (5).

$$yR_S \stackrel{?}{=} Q_S + hP_{pub1} \tag{5}$$

If the above equation satisfies, S assures that D_S is valid and properly generated by the KGC.
- Set-Secret-Value: Signer S selects a random integer x_S as the secret value.
- Set-Public-Key: The signer S computes $\widetilde{Q_S} = x_S R_S$ and sets the public key $PK_S = (R_S, Q_S, \widetilde{Q_S})$.
- CLS-Sign: Given a message m, *params*, (x_S, D_S) and PK_S, signatory S computes

$$T = tR_S \tag{6}$$

$$v = H(m \oplus ID_S, T, Q_S) \tag{7}$$

$$\tau = x_S[t + vx_S + y]^{-1} \bmod q \tag{8}$$

where $t \in_R Z_q^*$. After that, S sends the signature $\sigma = (\tau, T)$ to the verifier.
- CLS-Verify: Given *params*, m, σ, ID_S and PK_S, the verifier calculates

$$h = H(ID_S, R_S, P)$$
$$v = H(m \oplus ID_S, T, Q_S)$$

Next, the verifier checks the authenticity of σ as

$$\widetilde{Q_S} \stackrel{?}{=} \tau R' \tag{9}$$

$$\text{where } R' = T + (v\widetilde{Q_S}) + (hP_{pub1}) + Q_S \tag{10}$$

If Eq. (9) holds, then σ is treated as genuine and thus, it returns VALID; otherwise, it returns INVALID.

4. Security analysis and performance evaluation

This section demonstrates that the proposed CLS scheme is existentially unforgeable by the Type-I and Type-II adversaries ($\mathcal{A}_I$ and $\mathcal{A}_{II}$) under the hardness of ECDLP. In addition, a comparative study of proposed CLS with other CLS schemes is given later of this section Figs. 1 and 2.

4.1. Security analysis of the proposed CLS scheme

Claim 1. *Our CLS scheme is consistent, i.e., CLS-Verify algorithm always confirms the authenticity with a well-formed signature produced by a valid signer.*

Proof. The verifier runs the CLS-Verify algorithm to check the authenticity of $\sigma = (\tau, T)$. Now, anyone who has the access of message m can easily compute h and l using Eqs. (2) and (7) respectively, and outputs $\widetilde{R} = \tau R'$ as

Signer S	Secure channel	KGC
Set-Partial-Private-Key		
Identity: ID_S	$\xrightarrow{ID_S}$	
		Choose $r \in_R Z_q^*$
		Compute $R_S = rP$ and $Q_S = rl^{-1}P$
		Compute $h = H(ID_S, R_S, P)$
		Compute $y = \left[l^{-1} + \left(\frac{l}{r}\right)h\right] \mod q$
	$\xleftarrow{D_S = (y, R_S, Q_S)}$	
Check $yR_S \stackrel{?}{=} Q_S + hP_{pub1}$		
Set-Secret-Value		
Choose $x_S \in_R Z_q^*$		
Set x_S as secret value		
Set-Public-Key		
Compute $\tilde{Q}_S = x_S R_S$		
Set $PK_S = (Q_S, \tilde{Q}_S)$ as public key		

Fig. 1. Different key setting phases in our CLS Scheme.

Signer S	Public channel	Verifier V
CLS-Sign		
Choose $t \in_R Z_q^*$		
Compute $T = tR_S$ and $v = H(m \oplus ID_S, T, Q_S)$		
Compute $\tau = x_S\left[t + vx_S + y\right]^{-1} \mod q$		
	$\xrightarrow{\sigma = (\tau, T)}$	
		CLS-Verify
		Compute $h = H(ID_S, R_S, P)$
		Compute $l = H(m \oplus ID_S, T, Q_S)$
		Compute $R' = T + (v\tilde{Q}_S) + (hP_{pub1}) + Q_S$
		If $(\tilde{Q}_S \stackrel{?}{=} \tau R')$, accept $\sigma = (\tau, T)$
		Else, reject $\sigma = (\tau, T)$

Fig. 2. CLS-Sign and CLS-Verify phases in the proposed CLS scheme.

$$
\begin{aligned}
R' &= T + v\tilde{Q_S} + hP_{pub1} + Q_S, \text{[by Eqn. (10)]} \\
&= T + v\tilde{Q_S} + \left(\frac{l}{r}\right)hR_S + l^{-1}R_S \\
&= T + v\tilde{Q_S} + \left[l^{-1} + \left(\frac{l}{r}\right)h\right]R_S, \text{[by Eqns. (3)\&(6)]} \\
&= tR_S + vx_SR_S + yR_S \\
&= [t + vx_S + y]R_S, \text{[by Eqn. (8)]}
\end{aligned}
$$

Therefore, $\tau R' = x_S R_S = \tilde{Q_S}$. It may be noted that if the message m is altered, then Eq. (9) will never satisfy. Hence, the theorem follows. □

Theorem 1. *If the hash function is modeled as a random oracle, then the proposed CLS scheme is existentially unforgeable under adaptive chosen-message and -ID attacks based on the intractability of ECDLP.*

Proof. The theorem follows when Lemmas 2 and 3 according to Definitions 1 and 2 are satisfied. □

Lemma 2 (Type-I security)**.** *If there is a forger $\mathcal{A}_I$ that breaks (T, n_c, n_s, ϵ)−Type-I security of proposed CLS scheme in **Game 1**, then there exists an solver $\mathcal{F}_I$ that can breach ECDLP in elliptic curve group G_q with success probability ϵ' in polynomial time T' for*

$$\epsilon' \geq \left(1 - \frac{n_H(n_{KE} + n_{SV})}{q}\right)\left(1 - \frac{n_S}{q}\right)\left(\frac{1}{n_H}\right)\epsilon$$

$$T' = T + \mathcal{O}\Big((n_C + n_S)\ T_S\Big)$$

where n_H, n_{KE}, n_{SV}, n_S, n_C and T_S represent the number of Hash, Partial-Private-Key, Set-Public-Key, CLS-Sign, Create User queries and time taken for a single scalar point multiplication, respectively.

Proof. If the proposed CLS scheme is vulnerable, then there exists a Type-I forger $\mathcal{A}_I$ to breach the ECDLP. Suppose, the challenger $\mathcal{C}$ gives an ECDLP instance $\psi = \langle P, P_{pub}\rangle$ to the $\mathcal{A}_I$ for which $\mathcal{F}_I$ computes s that satisfies the condition of $P_{pub} = sP$, where P is the generator of G_q.

- Setup: Once $\mathcal{A}_I$ receives the challenge tuple ψ, it creates the following lists $\mathcal{L}$ of tuple $(ID, R, Q, \widetilde{Q}, x, y, h)$, $\mathcal{L}_H$ of tuple $(entry, R, Q, \widetilde{Q}, T, h, choice)$ and $\mathcal{R}$ of tuple $(ID, r, x, R, Q, \widetilde{Q})$. Initially, $\mathcal{L}_H$, $\mathcal{L}$, and $\mathcal{R}$ are empty lists. After that, $\mathcal{A}_I$ publishes the global system parameters $params = \big(G_q, G_m, q, P, P_{pub}, H, H_1, H_2\big)$. Now, $\mathcal{A}_I$ runs an algorithm $\mathcal{F}_I$ to solve the ECDLP.
- Create User: Suppose $\mathcal{A}_I$ asks a Create User query on the identity ID_i. $\mathcal{F}_I$ chooses $r, a, b \in_R Z_q^*$ and executes the following tasks:
  - computes $R_i = rP$.
  - sets $y = a$ and $h = H(ID_i, R_i, P) = -b \mod q$, and then computes $Q_i = yR_i + hP_{pub1}$.
  - stores $(ID_i, R_i, Q_i, \perp, 0, y, h)$ to $\mathcal{L}$ and $(ID_i, R_i, Q_i, \perp, \perp, h, 1)$ to $\mathcal{L}_H$.

  Note that the Partial-Private-Key verification condition $yR_i \stackrel{?}{=} Q_i + hP_{pub1}$ always holds.
- Hash Query: In this phase, $\mathcal{A}_I$ makes at most n_H number of H queries and each of them either of q_{ID} or q_m. $\mathcal{F}_I$ proceeds as
  - For query $q_{ID_j} = (ID_j, R_j, P)$, if $entry = ID_j$ and $choice = 1$ exist in $\mathcal{L}_H$, then it returns h to $\mathcal{A}_I$, otherwise, it executes Set-Partial-Private-Key query and returns h from $\mathcal{L}_H$ to $\mathcal{A}_I$.
  - For query $q_m = (m, ID_j, T, Q_j)$, if the entry exists with $choice = 0$, then it returns h to $\mathcal{A}_I$, otherwise, it chooses $h \in_R Z_q^*$ at random, sets $h \leftarrow H(m \oplus ID_j, T, Q_j)$ and inserts $(m \oplus ID_j, R_j, Q_j, \perp, T, h, 0)$ into $\mathcal{L}_H$. Finally, h is the output to $\mathcal{A}_I$.
- Set-Partial-Private-Key Query: If $\mathcal{A}_I$ makes the query on its selected ID_j and if it is found in $\mathcal{L}$, then $\mathcal{F}_I$ returns $D_j = (y, R_j, Q_j)$. Otherwise, $\mathcal{F}_I$ calls Create User for $ID_j \neq ID^*$ and outputs $D_j = (y, R_j, Q_j)$.
- Set-Secret-Value Query: Assume that $\mathcal{A}_I$ makes this query on its chosen ID_j. For $ID_j = ID^*$, $\mathcal{F}_I$ aborts the simulation. Otherwise, it checks an entry for ID_j $(\neq ID^*)$ in $\mathcal{L}$. If it exists (other than $\perp$), then $\mathcal{F}_I$ returns x; otherwise, it chooses $c_j \in_R Z_q^*$. Now, if an entry exists for $x_j = \perp$, then updates only x_j as $x_j = c_j$; otherwise, calls Create User, and then updates $x_j = c_j$ in $\mathcal{L}$.
- Set-Public-Key Query: The $\mathcal{A}_I$ invokes the query on its chosen ID_j and if it is found in $\mathcal{L}$, then $\mathcal{F}_I$ returns $PK_j = (Q_j, \widetilde{Q}_j)$. Otherwise, $\mathcal{F}_I$ calls Create User and proceeds with $ID_j \neq ID^*$ as
  - It checks an entry for ID_j in $\mathcal{L}$ where $x_j \neq \perp$. If such x_j exists, then sets $c = x_j$; otherwise, it chooses $c_j \in_R Z_q^*$. After that, it computes $\widetilde{Q}_j = c_j R_j$.
  - After that, it replaces $(\perp, 0)$ by $(\widetilde{Q}_j, c_j)$ of the corresponding tuple in $\mathcal{L}$, i.e., $(ID_j, R_j, Q_j, \widetilde{Q}_j, c_j, y, 1)$ to $\mathcal{L}$. Also, it updates $(\perp, \perp)$ by $(\widetilde{Q}_j, \perp)$ of the corresponding tuple in $\mathcal{L}_H$, i.e., $(ID_j, R_j, Q_j, \widetilde{Q}_j, \perp, h, 1)$ to $\mathcal{L}_H$.

  Finally, $\mathcal{F}_I$ sends $PK_j = (Q_j, \widetilde{Q}_j)$ to $\mathcal{A}_I$.
- Replace-Public-Key Query: Now, for invoked query $(ID_j, PK' = (R', Q', \widetilde{Q}'))$ where $\widetilde{Q}' = x'r'P$, $\mathcal{F}_I$ selects $\alpha \in_R Z_q^*$ and sets $y = \perp$, $h = \alpha$, $R = r'P$, $Q = Q'$, $\widetilde{Q} = \widetilde{Q}'$ and $x = x'$ which reflects in $\mathcal{L}$. Finally, $\mathcal{F}_I$ inserts $(ID, r', x', R', Q', \widetilde{Q}')$ to list $\mathcal{R}$.
- CLS-Sign Query: On receiving the query on $q_S = (ID_j, m)$ asked by $\mathcal{A}_I$, $\mathcal{F}_I$ checks for an entry in $\mathcal{R}$, if it does not exist, then generates the signature using the procedures given in Section 3.1. Otherwise, $\mathcal{F}_I$ chooses $a, b \in_R Z_q^*$ and sets $\tau = a$, $l = H(m \oplus ID_j, T, Q_j) = b$ and computes $T = (a^{-1} - l)xR_j - Q_j - hP_{pub1}$. Finally, $\mathcal{F}_I$ inserts $(m \oplus ID_j, R_j, Q_j, \widetilde{Q}_j, T, l, 0)$ to $\mathcal{L}_H$. $\mathcal{F}_I$ returns $\sigma_{q_S} = (\tau, T)$ to $\mathcal{A}_I$.
- Output: $\mathcal{A}_I$ stops asking queries and returns a forged signature $\sigma' = (\tau', T')$ for message m' and identity ID' whose public key is $PK' = (R', Q', \widetilde{Q}')$ where CLS-Verify$(params, m', \sigma', ID', PK') =$ VALID. Now, if $ID' \neq ID^*$, then $\mathcal{F}_I$ returns INVALID and stops the simulation. Otherwise, $\mathcal{F}_I$ retrieves tuple $(ID, R, Q, \widetilde{Q}, x, y, h)$ from $\mathcal{L}$ and $(ID, r, x, R, Q, \widetilde{Q})$ from $\mathcal{R}$. Now, by using the forking lemma [22], if we replay $\mathcal{F}_I$ with the unchanged random tape for different choices of h (from list $\mathcal{L}_H$), then two more valid signatures $\sigma_1' = (\tau_1', T)$ and $\sigma_2' = (\tau_2', T)$ can be produced.

$$\widetilde{Q} = \tau_i\big(T + (l_i\widetilde{Q}) + (hP_{pub1}) + Q\big),\ \ i \in [1, 3] \tag{11}$$

Now, for $i = \{1, 2, 3\}$, Eq. (11) can be written as

$$\begin{aligned} xR &= \tau_i\big(tR + l_i xR + hsP + rs^{-1}P\big) \\ xrP &= \tau_i\big(tr + l_i xr + hs + rs^{-1}\big)P \end{aligned} \tag{12}$$

Using Eq. (12), $\mathcal{F}_I$ has

$$xr = \tau_i\big(tr + l_i xr + hs + rs^{-1}\big),\ \ \forall i \in [1, 3] \tag{13}$$

Let, $A_i = (l_i\tau_i x - x)$, $B_i = h\tau_i$, $\alpha = tr$, $\beta = s^{-1}r$, then Eq. (13) can be written as

$$\tau_i(\alpha + \beta) + A_i r + B_i s = 0, \quad \forall\ i \in [1, 3] \tag{14}$$

By assuming $\gamma = \alpha + \beta$, we can rewrite Eq. (14) as

$$\tau_i\gamma + A_i r + B_i s = 0, \quad \forall i \in [1, 3] \tag{15}$$

Here, in Eq. (15), $\mathcal{F}_I$ can compute A_i and B_i with the help of lists $\mathcal{L}$ and $\mathcal{R}$. Now, $\mathcal{F}_I$ can solve the above three linearly independent equations to find out the unknowns γ, r, s. Therefore, $\mathcal{F}_I$ outputs s as the solution of elliptic curve discrete logarithm problem for P_{pub}. In addition, $\mathcal{F}_I$ can check the validity of s if required as

- Calculates $\beta = s^{-1}r$
- Calculates $t = r^{-1}(\gamma - \beta)$
- Checks $tR \stackrel{?}{=} T$ and $sP \stackrel{?}{=} P_{pub}$

Analysis of the Breaching Probability: For the successful forgery, there are three events as follows:

ϑ_1 : $\mathcal{F}_I$ executes sucessfully during the above simulation.
ϑ_2 : σ^* is a valid forged signature for (m^*, ID).
ϑ_3 : Forged signature σ^* fulfils the condition of $ID = ID^*$.

Therefore, the overall successful probability of breaking ECDLP in this Game 1 is defined as

$$\Pr[\vartheta_1 \wedge \vartheta_2 \wedge \vartheta_3] = \Pr[\vartheta_1] \cdot \Pr[\vartheta_2|\vartheta_1] \cdot \Pr[\vartheta_3|\vartheta_1 \wedge \vartheta_2]$$

- Simulation of the Set-Partial-Private-Key query stops if the Create User query fails. Also, Create User query aborts if the assignment of oracle $H(ID_i, R_i, P)$ is inconsistent. Now, this occurs with probability at most $\frac{n_H}{q}$. Therefore, the simulation for n_{KE} times is successful with probability at least $\left(1 - \frac{n_H}{q}\right)^{n_{KE}}$.
- $\mathcal{F}_I$ executes during the simulation of Set-Secret-Value query with success probability at least $\left(1 - \frac{n_H}{q}\right)^{n_{SV}}$.
- CLS-Sign query successfully produces a signature to train $\mathcal{A}_I$ for $ID \neq ID^*$ with probability $\left(1 - \frac{1}{q}\right)$. Therefore, it does not abort for n_S times with probability at least $\left(1 - \frac{1}{q}\right)^{n_S}$.
- It is assumed that probability of forging a valid signature for an identity chosen by $\mathcal{F}_I$ is ϵ.
- $\mathcal{A}_I$ succeeds to forge a signature for $ID = ID^*$ with probability $\frac{1}{n_H}$ with a restriction that CLS-Sign query does not abort.

Therefore, we have

$$\begin{aligned}\Pr[\vartheta_1] &= \left(1 - \frac{n_H}{q}\right)^{(n_{KE}+n_{SV})}\left(1 - \frac{1}{q}\right)^{n_S} \\ &\geq \left(1 - \frac{n_H(n_{KE}+n_{SV})}{q}\right)\left(1 - \frac{n_S}{q}\right)\end{aligned} \tag{16}$$

$$\Pr[\vartheta_2|\vartheta_1] \geq \epsilon \tag{17}$$

$$\Pr[\vartheta_3|\vartheta_1 \wedge \vartheta_2] \geq \left(\frac{1}{n_H}\right) \tag{18}$$

Hence, form Eqs. (16), (17) and (18), $\mathcal{F}_I$ has the overall success probability of breaking ECDLP

$$\begin{aligned}\Pr[\vartheta_1 \wedge \vartheta_2 \wedge \vartheta_3] &= \Pr[\vartheta_1] \cdot \Pr[\vartheta_2|\vartheta_1] \cdot \Pr[\vartheta_3|\vartheta_1 \wedge \vartheta_2] \\ \epsilon' &\geq \left(1 - \frac{n_H(n_{KE}+n_{SV})}{q}\right)\epsilon\left(\frac{1}{n_H}\right)\left(1 - \frac{n_S}{q}\right) \\ \text{or, } \epsilon' &\geq \left(1 - \frac{n_H(n_{KE}+n_{SV})}{q}\right)\left(1 - \frac{n_S}{q}\right)\left(\frac{1}{n_H}\right)\epsilon\end{aligned}$$

Analysis of the Running Time: Since, the execution cost of the scalar point multiplication on elliptic curve is a dominant operation than the other cryptographic operations in this simulation, we mainly consider the running cost T_S of this operation during the simulation of the above game. The algorithm $\mathcal{F}_I$ requires $2n_cT_S$, n_PT_S and $2n_sT_S$ during Set-Partial-Private-Key, Set-Public-Key and CLS-Sign queries, respectively. Therefore, the additional time required by $\mathcal{F}_I$ is $T_{add} = (2n_c + n_p + 2n_s)T_S \approx \mathcal{O}((n_c + n_s)T_S)$. Hence, the overall time needed by $\mathcal{F}_I$ to break ECDLP is considered as $T' = T + \mathcal{O}((n_c + n_s)T_S)$.

As ϵ is referred as a non-negligible advantage, so, $\mathcal{F}_I$ cannot breach the hardness of ECDLP. Therefore, our CLS scheme resists Type-I attack based on the random oracle model under the intractability assumption of the ECDLP. This completes the proof of Lemma 2. □

Lemma 3 (Type-II security). *If there is a forger $\mathcal{A}_{II}$ which breaks (T, n_E, n_S, ϵ)−Type-II security of proposed CLS scheme in **Game 2** model, then there exists a solver $\mathcal{F}_{II}$ that can breach ECDLP in elliptic curve group G_q with success probability ϵ' in polynomial time T' for*

$$\epsilon' \geq \left(1 - \frac{n_H n_{SV}}{q}\right)\left(1 - \frac{n_S}{q}\right)\left(\frac{1}{n_H}\right)\epsilon$$

$$T' = T + \mathcal{O}\Big((n_C + n_S)\ T_S\Big)$$

where n_H, n_{SV}, n_S n_C and T_S represent the number of Hash, Set-Public-Key, CLS-Sign, Create User queries and time taken for a single scalar point multiplication, respectively.

Proof. If the proposed CLS scheme is insecure, then a Type-II forger $\mathcal{F}_{II}$ exists that can breach the ECDLP. Suppose, the challenger $\mathcal{C}$ gives an ECDLP instance $\psi = \langle P, \Phi\rangle$ to the $\mathcal{F}_{II}$ for which $\mathcal{F}_{II}$ wants to find the unknown ϕ satisfying the condition $\Phi = \phi P$.

- Setup: On receiving the challenge tuple ψ, $\mathcal{F}_{II}$ creates the following lists $\mathcal{L}$ of tuple $(ID, R, Q, \widetilde{Q}, x, y, h)$ and $\mathcal{L}_H$ of tuple $(entry, R, Q, \widetilde{Q}, T, h, choice)$. The above mentioned two lists are initially empty. After that, $\mathcal{F}_{II}$ chooses $s \in_R Z_q^*$ and computes $P_{pub} = sP$. Finally, it publishes global system parameters $params = (G_q, G_m, q, P, P_{pub}, H, H_1, H_2)$. Now, $\mathcal{A}_{II}$ uses $\mathcal{F}_{II}$ in order to find a solution to the ECDLP instance.
- Create User: Assume that $\mathcal{A}_{II}$ asks a Create User query on the identity ID_j, and $\mathcal{F}_{II}$ computes several steps as follows:
  - It selects $a, b \in_R Z_q^*$, and defines $h = H(ID_j, R_j, P) = b$.
  - If $ID_j = ID^*$, then performs
    1. sets $y = a$, and $R_j = \Phi$.
    2. computes $Q_j = a\Phi - bP_{pub}$.
  - Else, i.e., $ID_j \neq ID^*$, performs
    1. computes $R_j = aP$ and $Q_j = s^{-1}R_j$.
    2. computes $y = (s^{-1} + a^{-1}sb) \mod q$.

  Finally, it stores $(ID_j, R_j, Q_j, \perp, \perp, y, h)$ to $\mathcal{L}$ and $(ID_j, R_j, Q_j, \perp, \perp, h, 1)$ to $\mathcal{L}_H$, and outputs $D_j = (y, R_j, Q_j)$.
- Hash Query: $\mathcal{A}_{II}$ makes at most n_H number of H queries and each of them either of q_{ID} or q_m. Now, $\mathcal{F}_{II}$ proceeds as:
  - For query $q_{ID} = (ID_j, R_j, P)$, if $entry = ID_j$ and $choice = 1$ exists in $\mathcal{L}_H$, then it outputs h to $\mathcal{A}_{II}$, otherwise, calls Set-Partial-Private-Key and outputs h from $\mathcal{L}_H$ to $\mathcal{A}_{II}$.
  - For query $q_m = (m, ID_j, T, Q_j)$, if the entry exists with $choice = 0$, then it outputs h to $\mathcal{A}_{II}$, otherwise, chooses $h \in_R Z_q^*$ at random, sets $h \leftarrow H(m \oplus ID_j, T, Q_j)$ and inserts $(m \oplus ID_j, R_j, Q_j, \perp, T, h, 0)$ into $\mathcal{L}_H$. Finally, h is returned to $\mathcal{A}_{II}$.
- Set-Partial-Private-Key Query: $\mathcal{A}_{II}$ asks queries on its chosen ID_j and if it is found in $\mathcal{L}$, then $\mathcal{F}_{II}$ returns $D_j = (y, R_j, Q_j)$. Otherwise, it simulates Create User query and outputs $D_j = (y, R_j, Q_j)$. It is noted that, the distribution is uniform, i.e., the underline condition $yR_j \stackrel{?}{=} Q_j + hP_{pub}$ always holds.
- Set-Secret-Value Query: Assume that $\mathcal{A}_{II}$ makes this query on its chosen ID_j. For $ID_j \neq ID^*$, $\mathcal{F}_{II}$ checks $\mathcal{L}$. If it exists (other than $\perp$), then $\mathcal{F}_{II}$ returns x_j; otherwise, it chooses $c_j \in_R Z_q^*$. Now, if an entry exists for $x_j = \perp$ then updates only x_j as $x_j = c_j$; otherwise, simulates Create User query and then updates $x_j = c_j$ in $\mathcal{L}$.
- Set-Public-Key Query: $\mathcal{A}_{II}$ invokes the query on its chosen ID_j and if it is found in list $\mathcal{L}$, then $\mathcal{F}_{II}$ returns $PK_j = (R_j, Q_j, \widetilde{Q}_j)$. Otherwise, $\mathcal{F}_{II}$ simulates Create User query and performs
  - It checks an entry for ID_j where $x_j \neq \perp$. If such x_j exists, then sets $c_j = x_j$; otherwise, it chooses $c_j \in_R Z_q^*$. After that, it computes $\widetilde{Q}_j = cR_j$.
  - It replaces $(\perp, \perp)$ by $(\widetilde{Q}_j, c_j)$ of the corresponding tuple in $\mathcal{L}$, i.e., $(ID_j, R_j, Q_j, \widetilde{Q}_j, c_j, y, 1)$ to $\mathcal{L}$. Also, it updates $(\perp, \perp)$ by $(\widetilde{Q}_j, \perp)$ of the corresponding tuple in $\mathcal{L}_H$, i.e., $(ID_j, R_j, Q_j, \widetilde{Q}_j, \perp, h, 1)$ to $\mathcal{L}_H$.

  Finally, $\mathcal{F}_{II}$ sends $PK = (R_j, Q_j, \widetilde{Q}_j)$ to $\mathcal{A}_{II}$.
- CLS-Sign Query: On receiving the query on $q_S = (ID_j, m)$ asked by $\mathcal{A}_{II}$, $\mathcal{F}_{II}$ checks whether $ID_j \stackrel{?}{=} ID^*$ holds. If it holds, then aborts the simulation; otherwise, it executes the following steps:
  - Chooses $a, b \in_R Z_q^*$ and then sets $\tau = a$ and $l = H(m \oplus ID_j, T, Q_j) = b$.
  - Computes $T = (x(a^{-1} - b)R_j) - (yR_j)$ with the help of $\mathcal{L}$.

  Finally, $\mathcal{F}_{II}$ inserts $(m \oplus ID_j, R_j, Q_j, \widetilde{Q}_j, T, l, 0)$ to $\mathcal{L}_H$, and returns $\sigma_{q_S} = (\tau, T)$ to $\mathcal{A}_{II}$.
- Output: $\mathcal{A}_{II}$ stops asking the queries and generates a forged signature $\sigma' = (\tau', T')$ for a chosen message m' for the identity ID' whose public key is $PK' = (R', Q', \widetilde{Q}')$ where CLS-Verify$(params, m', \sigma', ID', PK')$ =VALID. Now, if $ID' \neq ID^*$, then $\mathcal{F}_{II}$ outputs INVALID and aborts the simulation. Otherwise, $\mathcal{F}_{II}$ retrieves a tuple $(ID, R, Q, \widetilde{Q}, x, y, h)$ from $\mathcal{L}$. Now, by using the forking lemma [22], if we replay $\mathcal{F}_{II}$ with the unchanged random tape for different choices of h (from $\mathcal{L}_H$),

then two more valid signatures $\sigma_1' = (\tau_1', T)$ and $\sigma_2' = (\tau_2', T)$ can be produced.

$$\widetilde{Q} = \tau_i\big(T + (l_i\widetilde{Q}) + (hP_{pub1}) + Q\big),\ \ i \in [1,3] \tag{19}$$

It may be noted for $ID = ID'$, the assignment $Q = \Phi$ always hold. For $i \in [1, 3]$, the Eqn. (19) can be written as

$$\begin{aligned} xR &= \tau_i(t \cdot R + l_i xR + hsP + \phi P) \\ xrP &= \tau_i(tr + l_i xr + hs + \phi)P \end{aligned} \tag{20}$$

Using Eq. (20), $\mathcal{F}_{II}$ has

$$xr = \tau_i(tr + l_i xr + hs + \phi),\ \ \forall i \in [1,3] \tag{21}$$

Let, $A_i = (l_i\tau_i x - x)$, $B_i = h\tau_i$, $\alpha = tr$, then Eq. (21) can be written as

$$\tau_i\alpha + \tau_i\phi + A_i r = B_i s, \qquad \forall i \in [1,3] \tag{22}$$

In Eq. (22), $\mathcal{F}_{II}$ can compute A_i and B_i with the help of lists $\mathcal{L}$ and $\mathcal{R}$. It may be noted that s is known to $\mathcal{F}_{II}$. Now, $\mathcal{F}_{II}$ can solve the above three linearly independent equations to find out the unknowns α, ϕ, and r. Therefore, $\mathcal{F}_{II}$ outputs ϕ as the solution to ECDLP instance for Φ. In addition, $\mathcal{F}_{II}$ can check the validity of ϕ if required as

- Calculates $Q' = yR - hP_{pub}$ and checks $\phi P \stackrel{?}{=} Q'$

If the above mentioned condition holds, then it satisfies the requirement. It may be noted that if the whole process executes properly, then the condition will always hold.

Analysis of the Breaching Probability : For the successful forgery, there are three events as follows:

ϑ_1 : $\mathcal{F}_{II}$ executes successfully during the above simulation.
ϑ_2 : σ^* is a valid forged signature for (m^*, ID).
ϑ_3 : Forged signature σ^* fulfils the condition of $ID = ID^*$.

Therefore, the overall successful probability of breaking ECDLP in this Game 2 is defined as

$$\Pr[\vartheta_1 \wedge \vartheta_2 \wedge \vartheta_3] = \Pr[\vartheta_1] \cdot \Pr[\vartheta_2|\vartheta_1] \cdot \Pr[\vartheta_3|\vartheta_1 \wedge \vartheta_2]$$

- Simulation of Set-Secret-Value query fails if Create User query fails. Also, Create User query fails if the proposed oracle allocation $H(ID_j, R, P)$ is inconsistent. Now, this occurs with probability utmost $\frac{n_H}{q}$. Therefore, the simulation is successful for n_{SV} times with probability at least $(1 - \frac{n_H}{q})^{n_{KE}}$.
- CLS-Sign query successfully produces a signature to train $\mathcal{A}_{II}$ for $ID \neq ID^*$ with probability $(1 - \frac{1}{q})$. Therefore, it does not abort for n_S times with success probability at least $\left(1 - \frac{1}{q}\right)^{n_S}$.
- It is assumed that the probability of forging a valid signature for an identity by $\mathcal{F}_{II}$ is ϵ.
- $\mathcal{A}_{II}$ successfully forges a signature for $ID = ID^*$ with probability $\frac{1}{n_H}$ with a restriction that CLS-Sign query does not abort.

Therefore, we have

$$\begin{aligned} \Pr[\vartheta_1] &= \left(1 - \frac{n_H}{q}\right)^{n_{SV}}\left(1 - \frac{1}{q}\right)^{n_S} \\ \text{or, } \Pr[\vartheta_1] &\geq \left(1 - \frac{n_H n_{SV}}{q}\right)\left(1 - \frac{n_S}{q}\right) \end{aligned} \tag{23}$$

$$\text{Similarly, } \Pr[\vartheta_2|\vartheta_1] \geq \epsilon \tag{24}$$

$$\text{and } \Pr[\vartheta_3|\vartheta_1 \wedge \vartheta_2] \geq \left(\frac{1}{n_H}\right) \tag{25}$$

Therefore, form Eqs. (23), (24) and (25), $\mathcal{F}_{II}$ has the overall success probability of breaking ECDLP

$$\begin{aligned} \Pr[\vartheta_1 \wedge \vartheta_2 \wedge \vartheta_3] &= \Pr[\vartheta_1] \cdot \Pr[\vartheta_2|\vartheta_1] \cdot \Pr[\vartheta_3|\vartheta_1 \wedge \vartheta_2] \\ \epsilon' &\geq \left(1 - \frac{n_H n_{SV}}{q}\right)\epsilon\left(\frac{1}{n_H}\right)\left(1 - \frac{n_S}{q}\right) \\ \text{or, } \epsilon' &\geq \left(1 - \frac{n_H n_{SV}}{q}\right)\left(1 - \frac{n_S}{q}\right)\left(\frac{1}{n_H}\right)\epsilon \end{aligned}$$

Analysis of the Running Time: We mainly consider the cost of scalar point multiplication operation on elliptic curve during the execution of the above game. The algorithm $\mathcal{F}_{II}$ requires $2n_E T_S$, $n_P T_S$ and $2n_S T_S$ during Set-Partial-Private-Key,

Table 2
Computation cost of Primitive operations [15–17] .

Operation	Required time
T_i (Modular inversion)	$\approx 11.60\,T_m$
T_a (Addition of two elliptic curve points)	$\approx 00.12\,T_m$
T_s (Elliptic curve scalar point multiplication)	$\approx 29.00\,T_m$
T_e (Exponentiation)	$\approx 21.00\,T_m$
T_h (Map-To-Point hash function)	$\approx 29.00\,T_m$
T_p (Bilinear pairing)	$\approx 87.00\,T_m$

Table 3
Performance comparisons of related CLS schemes.

Scheme	Cost during signature		Total cost $(G+V)$		Signature length	Adversary		Hardness assumption
	generation (G)	verification (V)				Type-I	Type-II	
Gong and Li [9]	T_s	$4T_s+3T_a$	$5T_s+3T_a$	$\approx 145.36\,T_m$	$3\|G_q\|$	√	√	ECDLP
He et al. [12]	T_s+T_i	$3T_s+3T_a$	$4T_s+3T_a+T_i$	$\approx 127.96\,T_m$	$2\|G_q\|$	√	×	ECDLP
Islam and Biswas [16]	T_s	$4T_s+3T_a$	$5T_s+3T_a$	$\approx 145.36\,T_m$	$2\|G_q\|$	√	√	ECDLP
Tsai et al. [30]	T_s+T_i	$4T_s+3T_a$	$5T_s+3T_a+T_i$	$\approx 156.96\,T_m$	$2\|G_q\|$	√	×	ECDLP
Yeh et al. [35]	T_s	$4T_s+3T_a$	$5T_s+3T_a$	$\approx 145.36\,T_m$	$3\|G_q\|$	√	√	ECDLP
Zhang et al. [37]	$3T_s+2T_h+2T_a$	$4T_p+2T_h$	$4T_p+3T_s+4T_h+2T_a$	$\approx 551.24\,T_m$	$2\|G_q\|$	×	×	CDHP
Ours	T_s+T_i	$3T_s+3T_a$	$4T_s+3T_a+T_i$	$\approx 127.96\,T_m$	$2\|G_q\|$	√	√	ECDLP

Set-Public-Key and CLS-Sign queries, respectively. Therefore, additional time required by $\mathcal{F}_{II}$ is $T_{add}=(2n_E+n_P+2n_S)T_S\approx \mathcal{O}((n_E+n_S)T_S)$. Hence, the overall time needed by $\mathcal{F}_{II}$ to break ECDLP is considered as $T'=T+\mathcal{O}((n_E+n_S)T_S)$.

As ϵ is referred as a non-negligible advantage, so, $\mathcal{F}_{II}$ cannot breach the hardness of ECDLP. Therefore, the proposed CLS scheme resists Type-II attack under the random oracle model based on the hardness assumption of the ECDLP. This completes the proof of Lemma 3. □

4.2. Performance assessment and comparisons

In this section, we analyze the performance of our CLS against the security constrains, computational and communicational costs during signature generation and transmission, respectively.

4.2.1. Computational time

The cost of *Setup* algorithm comprises the generation of the group G_q and one scalar multiplication; Set-Partial-Private-Key algorithm needs to compute two scalar multiplications with one modular inverse operation; Set-Public-Key algorithm computes one scalar multiplication in group G_q. To calculate the signature σ for the message m, the signer runs CLS-Sign algorithm, which considers one modular inverse and one scalar point multiplication operation on the elliptic curve group. On the other hand, the verifier runs the CLS-Verify, which computes three elliptic curve scalar point multiplications, and three point addition operations to verify a signature. So, in total, our CLS scheme requires four scalar point multiplications, one modular inverse of an integer and three point addition operations, respectively. Based on [15–17], Table 2 is defined, which shows the computational cost required to perform each cryptographic operation, where T_m denotes the computation cost required to perform a modular multiplication operation. Table 3 shows a detailed performance comparison between ours and other CLS schemes where the required cost notations are defined in Table 2. In conformity with Table 3, it has been observed that the total cost of our CLS is nearly 88% of Gong and Li [9], 100% of He et al. [12], 88% of Islam and Biswas [16], 81% of Tsai et al. [30], 88% of Yeh et al. [35], and 23% of Zhang et al. [37] CLSs.

4.2.2. Signature length

In order to sign a message, the signer executes CLS-Sign algorithm and produces two-tuple signature $\sigma=(U,V)$, where $U\in Z_q^*$ and $V\in G_q$. Now, if we consider both are in the same order q, then the length of signature is $2|G_q|$, which is equal to the schemes in [12,16,30,37], and lesser than the schemes in [9,35].

In addition, Fig. 3 shows the communication cost during the signature generation and verification phases between different CLS schemes. Besides, our CLS scheme is computationally efficient than other paring-based CLS schemes that requires at least one pairing computation with either one elliptic curve scalar point multiplication or one MTP hash computation. Moreover, it is evident from the above discussion that only our CLS scheme achieves both the Type-I and Type-II securities with minimum computational overhead.

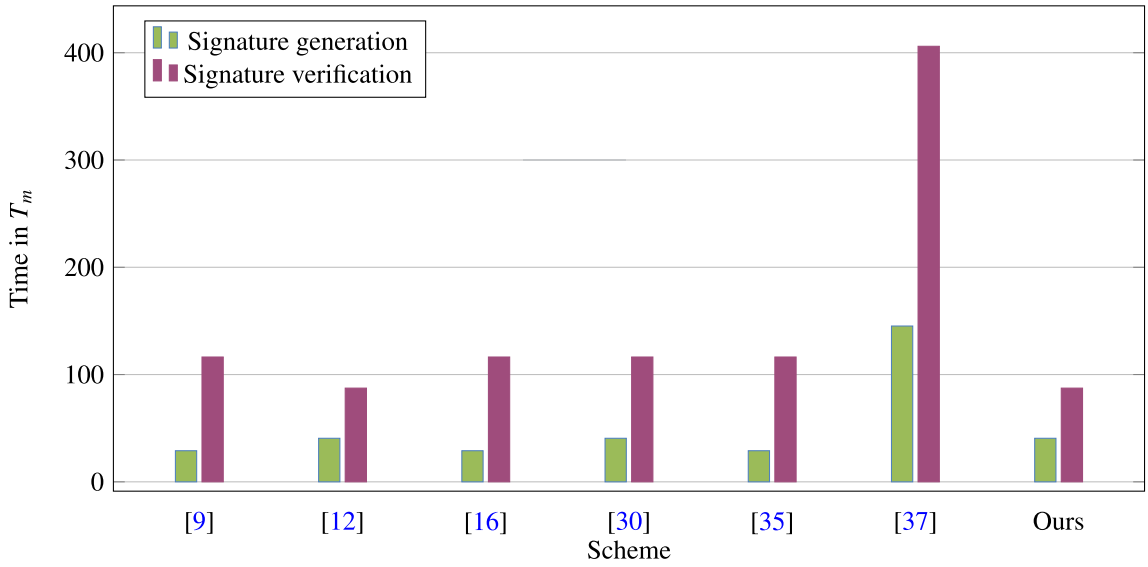

Fig. 3. Execution cost of the signature generation and verification of different CLS schemes.

5. Conclusion and future scope

An efficient and robust CLS scheme without using MTP hash function and bilinear pairing is presented in this paper. The proposed CLS is based on ECC and secure against Type-I and Type-II adversaries under the intractability assumption of the ECDLP and in the random oracle model. Also, the performance comparisons with other related CLS signature schemes demonstrated that our CLS scheme is computational efficient and has better security features. So, our CLS scheme is applicable in many scenarios, especially where the communication bandwidth and storage space are confined.

Although the proposed CLS is computationally efficient and has provable security, similar or different techniques can be developed without using random oracle in the future.

References

[1] S.S. Al-Riyami, K.G. Paterson, Certificateless Public Key Cryptography, in: Advances in cryptology-ASIACRYPT 2003, Springer, 2003, pp. 452–473.
[2] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: Proceedings of the 1st ACM conference on Computer and communications security, ACM, 1993, pp. 62–73.
[3] X. Cao, Ke.G. Paterson, W. Kou, An attack on a certificateless signature scheme., IACR Cryptology ePrint Archive 2006 (2006) 367.
[4] K.Y. Choi, J.H. Park, J.Y. Hwang, D.H. Lee, Efficient certificateless signature schemes, in: Applied Cryptography and Network Security, Springer, 2007, pp. 443–458.
[5] K. Choi, J. Park, D. Lee, A new provably secure certificateless short signature scheme, Comput. Math. Appl. 61 (7) (2011) 1760–1768.
[6] A. Dent, B. Libert, K. Paterson, Certificateless encryption schemes strongly secure in the standard model, in: International Workshop on Public Key Cryptography, Springer, 2008, pp. 344–359.
[7] H. Du, Q. Wen, Efficient and provably-secure certificateless short signature scheme from bilinear pairings, Comput. Standards Interf. 31 (2) (2009) 390–394.
[8] S. Feng, J. Mo, H. Zhang, Z. Jin, Certificateless short signature scheme from bilinear pairings, in: Applied Mechanics and Materials, 380, Trans Tech Publ, 2013, pp. 2435–2438.
[9] P. Gong, P. Li, Further improvement of a certificateless signature scheme without pairing, Int. J. Commun. Syst. 27 (10) (2014) 2083–2091.
[10] M. Gorantla, A. Saxena, An efficient certificateless signature scheme, in: International Conference on Computational and Information Science, 2005, pp. 110–116.
[11] D. Hankerson, A. Menezes, S. Vanstone, Guide to elliptic curve cryptography, Springer Science & Business Media, 2006.
[12] D. He, J. Chen, R. Zhang, An efficient and provably-secure certificateless signature scheme without bilinear pairings, Int. J. Commun. Syst. 25 (11) (2012) 1432–1442.
[13] D. He, B. Huang, J. Chen, New certificateless short signature scheme, IET Inf. Secur. 7 (2) (2013) 113–117.
[14] X. Huang, Y. Mu, W. Susilo, D. Wong, W. Wu, Certificateless signature revisited, in: Information Security and Privacy, Springer, 2007, pp. 308–322.
[15] S.H. Islam, G.P. Biswas, A pairing-free identity-based authenticated group key agreement protocol for imbalanced mobile networks, Ann. Telecommun. 67 (11–12) (2012) 547–558.
[16] S.H. Islam, G.P. Biswas, Provably secure and pairing-free certificateless digital signature scheme using elliptic curve cryptography, Int. J. Comput. Math. 90 (11) (2013) 2244–2258.
[17] S.H. Islam, M.S. Farash, G.P. Biswas, M.K. Khan, M.S. Obaidat, Provably secure and pairing-free certificateless digital multisignature scheme using elliptic curve cryptography, Int. J. Comput. Math. 90 (11) (2013) 2244–2258.
[18] A. Karati, G.P. Biswas, Cryptanalysis and improvement of a certificateless short signature scheme using bilinear pairing, in: Proceedings of the International Conference on Advances in Information Communication Technology & Computing, ACM, 2016, p. 19.
[19] N. Koblitz, Elliptic curve cryptosystems, Math. Comput. 48 (177) (1987) 203–209.
[20] V. Miller, Use of elliptic curves in cryptography, in: Conference on the Theory and Application of Cryptographic Techniques, 1985, pp. 417–426.
[21] D. Pointcheval, J. Stern, Security proofs for signature schemes, in: Eurocrypt, 96, 1996, pp. 387–398.
[22] D. Pointcheval, J. Stern, Security Proofs for Signature Schemes, Springer, Berlin, Heidelberg, pp. 387–398.

[23] A. Shamir, Identity-based Cryptosystems and Signature Schemes, in: Advances in Cryptology, Springer, 1984, pp. 47–53.
[24] K.-A. Shim, Breaking the short certificateless signature scheme, Inf. Sci. 179 (3) (2009) 303–306.
[25] Y. Sun, H. Li, Short-ciphertext and bdh-based cca2 secure certificateless encryption, Sci. China Inf. Sci. 53 (10) (2010) 2005–2015.
[26] Y.-X. Sun, F.-T. Zhang, Secure certificateless encryption with short ciphertext, Chin. J. Electron 19 (2) (2010) 313–318.
[27] M. Tian, L. Huang, Cryptanalysis of a certificateless signature scheme without pairings, Int. J. Commun. Syst. 26 (11) (2013) 1375–1381.
[28] M. Tian, L. Huang, W. Yang, On the security of a certificateless short signature scheme., IACR Cryptology 2011 (2011) 419. ePrint Archive
[29] J.-L. Tsai, A new efficient certificateless short signature scheme using bilinear pairings, IEEE Syst. J. 11 (4) (2015) 2395–2402.
[30] J.-L. Tsai, N.-W. Lo, T.-C. Wu, Weaknesses and improvements of an efficient certificateless signature scheme without using bilinear pairings, Int. J. Commun. Syst. 27 (7) (2014) 1083–1090.
[31] R. Tso, X. Huang, W. Susilo, Strongly secure certificateless short signatures, J. Syst. Softw. 85 (6) (2012) 1409–1417.
[32] R. Tso, X. Yi, X. Huang, Efficient and Short Certificateless Signature, in: Cryptology and Network Security, Springer, 2008, pp. 64–79.
[33] Z. Xu, X. Liu, G. Zhang, W. He, G. Dai, W. Shu, A certificateless signature scheme for mobile wireless cyber-physical systems, in: 2008 The 28th International Conference on Distributed Computing Systems Workshops, IEEE, 2008, pp. 489–494.
[34] W.-S. Yap, S.-H. Heng, B.-M. Goi, An Efficient Certificateless Signature Scheme, in: Emerging Directions in Embedded and Ubiquitous Computing, Springer, 2006, pp. 322–331.
[35] K.-H. Yeh, K.-Y. Tsai, C.-Y. Fan, An efficient certificateless signature scheme without bilinear pairings, Multimed. Tools Appl. 74 (16) (2015) 6519–6530.
[36] Z. Zhang, D. Feng, Key replacement attack on a certificateless signature scheme., IACR Cryptology ePrint Archive 2006 (2006) 453.
[37] Z. Zhang, D.S. Wong, J. Xu, D. Feng, Certificateless Public-key Signature: security Model and Efficient Construction, in: Applied Cryptography and Network Security, Springer, 2006, pp. 293–308.