Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(7): 12718–12730.
DOI: 10.3934/mbe.2023567
Received: 06 April 2023
Revised: 18 May 2023
Accepted: 22 May 2023
Published: 31 May 2023

Research article

Blockchain-enhanced certificateless signature scheme in the standard model

Xiaodong Yang[1,*], Haoqi Wen[1], Lei Liu[2], Ningning Ren[1] and Caifen Wang[3]

[1] College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China

[2] China Telecom WanWei Information Technology Co., LTD, Lanzhou 730030, China

[3] Department of Big Data and Internet, Shenzhen Technology University, Shenzhen 518118, China

* **Correspondence:** Email: y200888@163.com.

Abstract: The Internet of Things (IoT), driven by wireless communication and other technologies, is gradually entering our lives and promoting the transformation of society from "informatization" to "intelligence". Certificateless signature (CLS) eliminates the characteristic of certificate management, making it an effective method for verifying large-scale data in the IoT environment. Nevertheless, hash functions are regarded as ideal random oracles in the security proofs of most CLS schemes, which cannot guarantee the security of CLS schemes in reality. In response to this problem, Shim devised a CLS scheme without random oracles in the standard model and declared it to be provably secure. Unfortunately, in this paper, we cryptanalyze Shim's CLS scheme and demonstrate that it is not resistant to public key replacement attacks from a Type I attacker. Furthermore, to further improve the security of the Shim CLS scheme and avoid the single-point failure of the KGC and the signature forgery initiated, we propose a blockchain-based CLS scheme without a random oracle. Finally, we evaluate the comprehensive performance, and while maintaining the computational and communication performance of the Shim scheme, we resist both Type I and Type II attackers, as well as signature forgery initiated against public parameters.

Keywords: certificateless signature; forgery attack; random oracle model; blockchain; unforgeability

1. Introduction

The Internet of Things (IoT) connects items through sensors, controllers, and other devices to facilitate information exchange and communication in various application areas [1], such as environmental protection, intelligent transportation, public safety, food traceability, industrial monitoring, personal health, and intelligence collection. For example, smart transportation closely matches people, vehicles, and roads to improve traffic efficiency, ensure traffic safety, improve the traffic environment, and increase energy efficiency [2]. In environmental protection, it improves resource utilization, achieves energy saving and emission reduction [3]. However, with the increasing number of devices in the IoT, ensuring integrity verification and identity authentication among a large number of devices has become a critical and realistic issue [4].

To achieve effective authentication of the large amount of data transmitted in the IoT, ensuring data integrity, non-repudiation, and source identity authentication [5], Certificateless Signatures (CLS) is a commonly used solution, that avoids multiple algorithm parallel implementation to reduce efficiency [6]. In CLS, the user's signature key is created by combining a partial private key generated by the Key Generation Center (KGC) with the user's own secret key. Due to the independent operations of KGC and users, the CLS scheme effectively solves the problems of public key management and key custody. However, the design of CLS leads to it facing two types of attackers [7].

1). **Type I attacker** A_1: This is a type of attacker who impersonates a dishonest user. Specifically, A_1 can be qualified to have the user's secret key to initiate a public key replacement (PKR) attack, but the user's private key is kept secret to A_1.
2). **Type II attacker** A_2: Malicious KGC is portrayed as this type of attacker. Specifically, A_2 has all the functionality of KGC and launches a malicious-but-passive KGC (MBPK) attack. However, A_2 is prohibited from holding the user's secret key or replacing the user's public key.

To ensure the security of IoT devices, it is necessary to prevent these two types of attacks in the IoT environment.

IoT has gradually been combined with blockchain to address IoT security issues [8, 9]. The distributed nature of blockchain ensures that the data stored on the chain cannot be tampered with, thus solving trust issues and ensuring data security [10]. Therefore, in CLS, the user's partial private key is created through a blockchain smart contract, which avoids forgery attacks launched by attackers using public parameters.

The remainder of this paper is arranged as follows. Some preliminary knowledge related to our CLS scheme is presented in Section 3. Section 4 describes and cryptographically analyzes Shim's scheme [11]. Section 5 depicts the improved CLS scheme, and Section 6 analyzes its safety and effectiveness. Section 7 performs an analysis and comparison of the performance of the proposed scheme. Finally, the conclusion of this paper is summarized in Section 8.

2. Related work

Since the introduction of the concept of CLS by Ai-Riyami and Paterson [12], plenty of CLS schemes [13–15] have been designed. Existing CLS schemes can be broadly categorized into two types: under the random oracle model and under the standard model without the random oracle. While it is common and convenient to use the random oracle model to establish security proofs, it does not guarantee that the scheme will remain secure in the real world [16]. Hence, proving security in the standard model has become necessary. Most CLS schemes abstract cryptographic hash functions as ideal random oracles. However, such CLS schemes may have security vulnerabilities in practice. Therefore, designing a CLS scheme that does not rely on random oracles is more practical for authenticating IoT data integrity [17].

In recent years, CLS schemes without random oracles have attracted a great deal of attention from researchers. The first CLS scheme without random oracles was proposed by Liu et al. [18], but it was found to be vulnerable to MBPK attacks [19]. Subsequently, Yuan et al. [20] designed a CLS scheme that addressed MBPK attacks, but it was found to be vulnerable to PKR attacks [21]. Yu et al. [22] proposed another efficient CLS scheme, but it still could not resist MBPK and PKR attacks. To address the security issues in the Yu scheme [22], Yuan and Wang [23] proposed an improved scheme, but it is still not secure and can be vulnerable to MBPK attacks.

Also, Shim [11] devised a CLS scheme that did not require random oracles and proved that its security only depended on computing the unsolvability of the Diffie-Hellman (CDH) problem. Nevertheless, in this paper, we demonstrate that it is insecure against PKR attacks. A summary of our main work is given below.

1). We present an attack method against Shim's scheme [11]. Specifically, by substituting the user's public key, the legitimate signature of any desired message can be generated by A_1.
2). To remedy the security flaws of Shim's scheme [11], we present a blockchain-based CLS scheme without random oracles. Especially, our enhanced scheme utilizes the smart contract of the blockchain to represent the traditional KGC, which avoids the single-point failure of the KGC and the signature forgery initiated by the attacker with the help of public parameters.
3). We formally analyze the security of the improved scheme to show its ability to resist MBPK and PKR attacks.
4). We compare the performance of the improved solution with that of Shim's scheme [11] to illustrate the practical feasibility of our scheme.

3. Preliminaries

Table 1. Caption of the table.

Symbols	Description
G_1, G_2	Two elliptic curve groups
p	A prime number
g	A generator of group G_1, G_2
Z_p^*	A galois field
ID	The idntity of the user
m	The message to be signed
H_u, H_m	Two secure hash functions
msk	The system master key
$(psk^{(1)}, psk^{(2)})$	The private key
$(pk^{(1)}, pk^{(2)})$	The public key
σ	The signature on m

Table 1 shows the symbols used in this article. We choose two cyclic groups G_1 and G_2 and require their orders to be the same prime p. Next, we pick a generator g of G_1 and a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ satisfying the following conditions.

- Bilinear: For any $t_1, t_1 \in Z_p^*$, $e(g^{t_1}, g^{t_2}) = e(g, g)^{t_1 t_2}$.
- Non-degenerate: $e(g, g) \neq 1$.

- Computable: $e(g^{t_1}, g^{t_2})$ can be calculated efficiently.

In addition, (p, G_1, G_2, g, e) is commonly referred to as the bilinear group.

Several mathematical problems used in this article are described below, and they are difficult to solve in polynomial time.

- **CDH problem:** A triple (g, g^a, g^b) is known, where the unknown values $a, b \in Z_p^*$, and the CDH problem is to calculate g^{ab}.
- **Inverse-CDH problem:** Given a tuple (g, g^a), calculate g^{-a}, where $a \in Z_p^*$.
- **Discrete logarithm (DL) problem:** A tuple (g, g^a) is known and the DL problem is to compute $a \in Z_p^*$.

4. Attacks on Shim's CLS scheme

4.1. Review of Shim's CLS scheme

The construction of Shim's CLS scheme [11] is briefly described below.

- **Setup:** KGC executes the following to produce system parameters.
  1). Select the bilinear group (p, G_1, G_2, g, e) according to the selected security parameter ϑ.
  2). Choose $\alpha \in Z_p^*$ and calculate $g_1 = g^\alpha$.
  3). Choose $g_2, g_3 \in G_1$ and calculate $Z = e(g_1, g_2)$.
  4). Pick $u', \hat{u}_1, \cdots, \hat{u}_{n_u}$ from G_1 and set $\hat{U} = (\hat{u}_1, \cdots, \hat{u}_{n_u})$. Note that n_u is the byte length of the user identity.
  5). Pick $m', \hat{m}_1, \cdots, \hat{m}_{n_m}$ from G_1 and set $\hat{M} = (\hat{m}_1, \cdots, \hat{m}_{n_m})$, where n_m is the byte length of the message to be signed.
  6). Select two collision-resistant hash functions $H_u : \{0,1\}^* \to \{0,1\}^{n_u}$ and $H_m : \{0,1\}^* \to \{0,1\}^{n_m}$.
  7). Keep the master key $msk = g_2^\alpha$ secretly.
  8). Broadcast $params = (p, G_1, g, g_1, G_2, e, g_2, Z, g_3, u', m', \hat{U}, \hat{M}, H_u, H_m)$.
- **Partial-Private-Key-Extract:** Based on the user's identity ID, the user's partial private key $parkey_{ID}$ is produced by KGC.
  1). Calculate $u = H_u(ID)$.
  2). Define the ith bit of u to be $u[i]$, and assign the set of indices that satisfy $u[i] = 1$ to be $U \subset \{1, \cdots, n_u\}$.
  3). Define $\tilde{U} = u' \prod_{i \in U} \hat{u}_i$ as part of partial private key.
  4). Randomly select $r_u \in Z_p^*$, then calculate $psk^{(1)} = g_2^\alpha \cdot \tilde{U}^{r_u}$ and $psk^{(2)} = g^{r_u}$.
  5). Set $parkey_{ID} = (psk^{(1)}, psk^{(2)})$ as ID's partial private key.
- **User-Key-Generation:** Assuming that ID is the identity of the user, the user performs the following actions.
  1). Select $x, \tau \in Z_p^*$ at random and set its secret key $usk_{ID} = (x, \tau)$.
  2). Calculate $pk^{(1)} = g^\tau$ and $pk^{(2)} = g^x$.
  3). Set $upk_{ID} = (pk^{(1)}, pk^{(2)})$ as its public key.

- **CL-Sign:** For a message m, the user whose identity is ID performs the following signature steps.
  1). Calculate $\tilde{m} = H_m(m, ID, upk_{ID})$.
  2). Define the ith bit of $\tilde{m}$ to be $\tilde{m}[i]$, and assign the set of indices that satisfy $\tilde{m}[i] = 1$ to be $M \subset \{1, \cdots, n_m\}$.
  3). Define $\tilde{M} = m' \prod_{i \in M} \hat{m}_i$ as part of signature.
  4). Pick $k, r \in Z_p^*$ randomly, then calculate $\sigma_1 = \left[psk^{(1)} \cdot \tilde{U}^r \cdot g_3^x \cdot \tilde{M}^k\right]^{\tau^{-1}} = \left[g_2^\alpha \cdot \tilde{U}^{r+r_u} \cdot g_3^x \cdot \tilde{M}^k\right]^{\tau^{-1}}$, $\sigma_2 = g^r \cdot psk^{(2)} = g^{r+r_u}$ and $\sigma_3 = g^k$.
  5). Set m's signature as $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.
- **CL-Vfy:** For a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ on m from a user with ID, the verifier checks as follows.
  1). Calculate $u = H_u(ID)$.
  2). Based on ID's public key upk_{ID}, calculate $\tilde{m} = H_m(m, ID, upk_{ID})$.
  3). Verify the following equation:

$$e(pk^{(1)}, \sigma_1) = Z \cdot e(g_3, pk^{(2)}) \cdot e(\tilde{U}, \sigma_2) \cdot e(\tilde{M}, \sigma_3) \tag{4.1}$$

  4). If Eq (4.1) holds, σ is valid; otherwise, σ is invalid.

4.2. Weakness of Shim's CLS scheme

Shim [11] claims that their CLS scheme is secure against MBPK and PKR attacks. Nevertheless, we indicate that their scheme is not resistant to PKR attacks launched by Type I attackers. Type I attacker A_1 represents a malicious signer. The identity of the attacked user is assumed to be ID^*, although A_1 does not know the partial private key of ID^*, it can forge a valid signature for any message $m^* \neq m$ by replacing ID^*'s public key. This allows the forged message to pass signature verification. The forgery attack launched by A_1 is described in detail as follows.

1). A_1 picks $x^* \in Z_p^*$ at random and calculates $pk^{(2^*)} = g_1^{x^*}$.
2). A_1 sets $pk^{(1^*)} = g_1$.
3). A_1 assigns $upk_{ID}^* = (pk^{(1^*)}, pk^{(2^*)})$ as ID^*'s public key.
4). A_1 calculates $\tilde{m}^* = H_m(m^*, ID^*, upk_{ID}^*)$.
5). A_1 defines the ith bit of $\tilde{m}^*$ to be $\tilde{m}^*[i]$, and assign the set of indices that satisfy $\tilde{m}^*[i] = 1$ to be $M^* \subset \{1, \cdots, n_m\}$. Similarly, A_1 defines $U^* \subset \{1, \cdots, n_u\}$, where $u^* = H_u(ID^*)$.
6). A_1 defines $\tilde{U}^* = u' \prod_{i \in U^*} \hat{u}_i$ and $\tilde{M}^* = m' \prod_{i \in M^*} \hat{m}_i$
7). A_1 picks $k^*, r^* \in Z_p^*$ at random and calculates $\sigma_1^* = g_2 \cdot (\tilde{U}^*)^{r^*} \cdot g_3^{x^*} \cdot (\tilde{M}^*)^{k^*}$, $\sigma_2^* = g_1^{r^*}$ and $\sigma_3^* = g_1^{k^*}$.
8). A_1 sets a forged signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$ on m^*.

The following equation shows that the signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$ forged by A_1 is valid on m^* under ID^* with upk_{ID}^*.

$$\begin{aligned} e(pk^{(1^*)}, \sigma_1^*) &= e\left(g_2 \cdot (\tilde{U}^*)^{r^*} \cdot g_3^{x^*} \cdot (\tilde{M}^*)^{k^*}, g_1\right) \\ &= e(g_1, g_2)\, e\left((\tilde{U}^*)^{r^*}, g_1\right) e\left(g_3^{x^*}, g_1\right) e\left((\tilde{M}^*)^{k^*}, g_1\right) \end{aligned} \tag{4.2}$$

$$
\begin{aligned}
&= Z \cdot e\left(\tilde{U}^*, g_1^{r^*}\right) \cdot e\left(g_1^{x^*}, g_3\right) e\left(\tilde{M}^*, g_1^{k^*}\right) \\
&= Z \cdot e\left(g_3, pk^{(2*)}\right) \cdot e\left(\tilde{U}^*, \sigma_2^*\right) \cdot e\left(\tilde{M}^*, \sigma_3^*\right)
\end{aligned}
$$

From Eq (4.2), it can be seen that A_1 forged a valid signature. That is A_1's PKR attack is successful. Hence, Shim's CLS scheme [11] is not resistant to Type I attackers. In the IoT environment, a malicious signer can achieve identity authentication and pass forged messages through this method.

5. Blockchain-based CLS scheme without random oracles

Based on Shim's CLS scheme [11], we construct an improved CLS scheme using blockchain technology.

5.1. System model

The system model of our CLS scheme is depicted in Figure 1, involving four entities: the administrator, the smart contract-based KGC (SC-KGC), the user and the verifier.

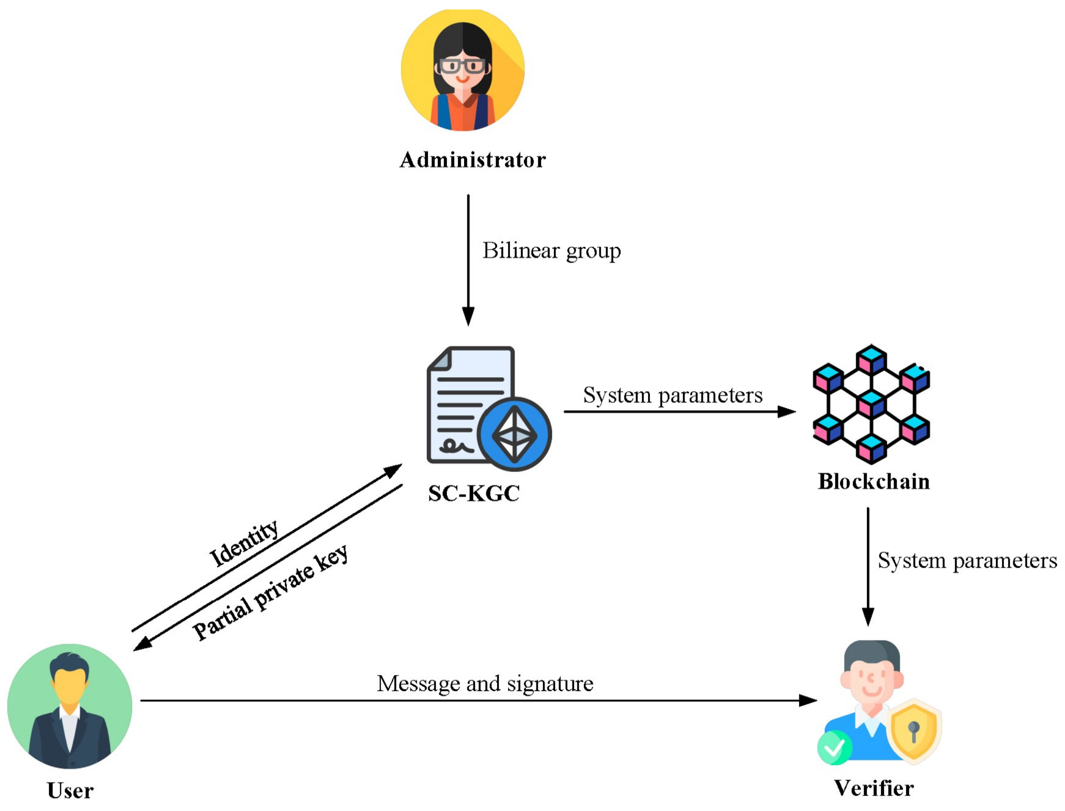

Figure 1. System model of the improved CLS scheme.

- **Administrator:** The administrator is primarily responsible for maintaining the blockchain network and initializing the system parameters.

- **SC-KGC:** This smart contract, deployed on the blockchain, mainly issues the user's partial private key and stores system parameters on the blockchain.
- **User:** Each user generates its public and private key, resulting in the certificateless signature of the message.
- **Verifier:** The verifier mainly verifies the validity of the signature generated by the user on the basis of the system parameters.

5.2. Improved CLS scheme

The proposed scheme is described in detail below.

- **Setup:** After receiving (p, G_1, G_2, g, e) sent by the administrator, SC-KGC executes as follows.
  1. SC-KGC selects $\alpha \in Z_p^*$ randomly and calculate $g_1 = g^{\alpha}$.
  2. SC-KGC selects $g_2 \in G_1$ and calculate $Z = e(g_1, g_2)$.
  3. SC-KGC stores the master key $msk = g_2^{\alpha}$ secretly.
  4. In the same way as the **Setup** algorithm in the Shim's scheme [11], SC-KGC generates the system parameters $params = (p, G_1, g, g_1, G_2, e, g_2, Z, u^{'}, m^{'}, \hat{U}, \hat{M}, H_u, H_m)$and broadcasts them on the blockchain.
- **Partial-Private-Key-Extract:** For the identity ID submitted by the user, SC-KGC executes as follows.
  1). Similar to Shim's scheme [11], SC-KGC calculates ID's partial private key $parkey_{ID} = (psk^{(1)}, psk^{(2)})$ and secretly transmits it to the user, where $psk^{(1)} = g_2^{\alpha} \cdot \left(\tilde{U}\right)^{r_u}$.
  2). SC-KGC uploads $psk^{(2)} = g^{r_u}$ to the blockchain.
- **User-Key-Generation:** Assuming that ID is the identity of the user, the user performs the following actions.
  1). Select $\tau \in Z_p^*$ at random and set its secret key $usk_{ID} = \tau$.
  2). Calculate $pk^{(1)} = g^{\tau}$ and $pk^{(2)} = \tilde{U}^{\tau}$.
  3). Set $upk_{ID} = (pk^{(1)}, pk^{(2)})$ as ID's public key .
- **CL-Sign:** For a message m, the user whose identity is ID performs the following steps.
  1). Calculate $\tilde{m} = H_m(m, ID, upk_{ID})$.
  2). Define the ith bit of $\tilde{m}$ to be $\tilde{m}[i]$, and assign the set of indices that satisfy $\tilde{m}[i] = 1$ to be $M \subset \{1, \cdots, n_m\}$.
  3). Define $\tilde{M} = m^{'} \prod_{i \in M} \hat{m}_i$ as part of signature.
  4). Pick $k, r \in Z_p^*$ at random, then calculate $\sigma_2 = g^r \cdot psk^{(2)} = g^{r+r_u}$ and $\sigma_3 = g^k$.
  5). Calculate $\sigma_1 = \left[psk^{(1)} \cdot \tilde{U}^r \cdot \tilde{M}^k\right]^{\tau^{-1}} = \left[g_2^{\alpha} \cdot \left(\tilde{U}\right)^{r+r_u} \cdot \tilde{M}^k\right]^{\tau^{-1}}$.
  6). Set the signature of m as $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.
- **CL-Vfy:** For a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ on m from a user with ID, the verifier checks as follows.
  1). Calculate $u = H_u(ID)$.
  2). Based on ID's public key upk_{ID}, calculate $\tilde{m} = H_m(m, ID, upk_{ID})$.

3). Verify the following equations:

$$e(g, pk^{(2)}) = e(\tilde{U}, pk^{(1)}) \tag{5.1}$$

$$e(pk^{(1)}, \sigma_1) = Z \cdot e(\tilde{U}, \sigma_2) \cdot e(\tilde{M}, \sigma_3) \tag{5.2}$$

4). If both Eqs (5.1) and (5.2) hold, the verifier accepts σ; otherwise, σ is rejected.

Correctness: Equation (5.1) is correct since

$$e(g, pk^{(2)}) = e\left(g, \tilde{U}^{\tau}\right) = e\left(\tilde{U}, g^{\tau}\right) = e\left(\tilde{U}, pk^{(1)}\right). \tag{5.3}$$

Equation (5.2) is correct since

$$\begin{aligned} e(pk^{(1)}, \sigma_1) &= e\left(g^{\tau}, \left[g_2^{\alpha} \cdot \left(\tilde{U}\right)^{r+r_u} \cdot \tilde{M}^k\right]^{\tau^{-1}}\right) \\ &= e\left(g, g_2^{\alpha} \cdot \left(\tilde{U}\right)^{r+r_u} \cdot \tilde{M}^k\right) \\ &= Z \cdot e(\tilde{U}, \sigma_2) \cdot e(\tilde{M}, \sigma_3). \end{aligned} \tag{5.4}$$

6. Security proof

Similar to Shim's CLS scheme [11], the improved CLS is proven to be secure by exploiting the security game between the attacker and the challenger. If the CDH problem is intractable in polynomial time, our improved CLS scheme is resistant to PKR attacks from Type I attackers.

Let A_1 be a Type I attacker who forges a legitimate signature of our CLS scheme with probability ε_1. Then, a challenger C_1 can successfully solve the CDH problem using A_1's forged signature. C_1 is assigned a CDH instance (g, g^a, g^b), and needs to interact with A_1 as follows in order to compute g^{ab}.

System initialization: Assume that the number of partial private key queries initiated by A_1 is C_{par} and the number of signature queries is C_s. C_1 initializes the following system parameters.

1). Set $l_u = 2(C_{par} + C_s)$ to satisfy $l_u(n_u + 1) \le p$.
2). Set $l_m = 2C_s$ to satisfy $l_m(n_m + 1) \le p$.
3). Choose two values k_u and k_m in $[0, n_u]$ and $[0, n_m]$, respectively.
4). Select $n_u + 1$ integers $\hat{x}, \hat{x}_1, \cdots, \hat{x}_{n_u}$ from $[0, l_u)$.
5). Select $n_m + 1$ integers $\hat{z}, \hat{z}_1, \cdots, \hat{z}_{n_m}$ from $[0, l_m)$.
6). Select $n_u + n_m + 2$ integers $\hat{y}, \hat{w}, \hat{y}_1, \cdots, \hat{y}_{n_u}, \hat{w}_1, \cdots, \hat{w}_{n_m}$ from $[0, p)$.
7). For $u = H_u(ID)$, define functions $F(u) = \hat{x} - k_u l_u + \sum_{i \in U} \hat{x}_i$ and $J(u) = \tilde{y} + \sum_{i \in U} \hat{y}_i$.
8). For $\tilde{m} = H_m(m, ID, upk_{ID})$, define two functions: $K(\tilde{m}) = \tilde{z} - k_m l_m + \sum_{i \in M} \tilde{z}_i$ and $L(\tilde{m}) = \tilde{w} + \sum_{i \in M} \tilde{w}_i$.
9). Set $g_1 = g^a$, $g_2 = g^b$, $u' = g_2^{\hat{x} - k_u l_u} g^{\hat{y}}$, $\hat{u}_i = g_2^{\hat{x}_i} g^{\hat{y}_i}$, $m' = g_2^{\hat{z} - k_m l_m} g^{\hat{w}}$, $\hat{m}_j = g_2^{\hat{z}_j} g^{\hat{w}_j}$, where $i \in [1, n_u]$ and $j \in [1, n_m]$.
Note that A_1 and C_1 cannot know the system master key $g_2^a = g^{ab}$. Furthermore, the following two equations hold:

$$\tilde{U} = g_2^{F(\tilde{u})} g^{J(\tilde{u})}, \quad \tilde{M} = g_2^{K(\tilde{m})} g^{L(\tilde{m})} \tag{6.1}$$

10). Send system parameter $params$ to A_1.

Create-User-Queries: C_1 creates a list L_U with an initial value of null. When A_1 requests ID_i's public key, C_1 passes upk_i to A_1 if L_U contains a tuple of ID_i. Otherwise, C_1 executes as follows.

1). Pick $\tau_i \in Z_p^*$ at random and calculate ID_i's secret key $usk_{ID_i} = \tau_i$.
2). Calculate $u_i = H_u(ID_i)$.
3). Define the set of indices that satisfy $u_i[j] = 1$ to be $U_i \subset \{1, \cdots, n_u\}$.
4). Define $\tilde{U}_i = u' \prod_{j \in U_i} \hat{u}_j$.
5). Calculate $pk_i^{(1)} = g^{\tau_i}$ and $pk_i^{(2)} = \tilde{U}_i^{\tau_i}$.
6). Set ID_i's public key $upk_{ID_i} = (pk_i^{(1)}, pk_i^{(2)})$.
7). If $F(u_i) = 0 mod p$, set $parkey_{ID_i} = (psk_i^{(1)}, psk_i^{(2)}) = (\perp, \perp)$; otherwise, select $r_{u_i} \in Z_p^*$, and then calculate $psk_i^{(1)} = g_1^{\frac{-J(u_i)}{F(u_i)}} \cdot \tilde{U}_i^{r_{u_i}}$ and $psk_i^{(2)} = g_1^{\frac{-1}{F(u_i)}} \cdot g^{r_{u_i}}$.
8). Add a tuple $(ID_i, usk_{ID_i}, upk_{ID_i}, psk_i^{(1)}, psk_i^{(2)})$ in L_U.
9). Transmit upk_i to A_1.

Partial-private-key-Queries: When A_1 asks for ID_i's partial private key, C_1 looks up the tuple $(ID_i, usk_{ID_i}, upk_{ID_i}, psk_i^{(1)}, psk_i^{(2)})$ in L_U and passes $parkey_{ID_i} = (psk_i^{(1)}, psk_i^{(2)})$ to A_1.

Secret-value-Queries: When A_1 asks for ID_i's secret value, C_1 looks up the tuple $(ID_i, usk_{ID_i}, upk_{ID_i}, psk_i^{(1)}, psk_i^{(2)})$ in L_U and passes usk_{ID_i} to A_1.

Replace-public-key-Queries: When A_1 wants to replace ID_i's public key with $upk_{ID_i}^*$, C_1 replaces upk_{ID_i} with $upk_{ID_i}^*$ in the list L_U.

Signature-Queries: When A_1 requests a query for messages m_j and ID_i, C_1 looks up the tuple $(ID_i, usk_{ID_i}, upk_{ID_i}, psk_i^{(1)}, psk_i^{(2)})$ in L_U and calculates $\tilde{m}_j = H_m(m_j, ID_i, upk_{ID_i})$.

1). If $F(u_i) \neq 0 \bmod p$, C_1 invokes the **CL-Sign** algorithm and passes the calculated signature to A_1.
2). If $F(u_i) = 0 \bmod p$ and $K(\tilde{m}_j) \neq 0 \bmod p$, C_1 picks $k, r \in Z_p^*$ randomly, then calculates $\sigma_{i2} = g^r$ and $g_1^{\frac{-1}{K(\tilde{m}_j)}} \cdot g^k$. Next, C_1 calculates

$$\sigma_{i1} = \left(\tilde{U}^r \cdot g_1^{\frac{L(\tilde{m}_j)}{K(\tilde{m}_j)}} \cdot \tilde{M}^k \right)^{\tau_i^{-1}}. \tag{6.2}$$

Finally, C_1 returns $(\sigma_{i1}, \sigma_{i2}, \sigma_{i3})$ to A_1.
3). Otherwise, C_1 terminates the game.

Forgery: A_1 forge a valid signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$ on a message m^* under an identity ID^*, where $u^* = H_u(ID^*)$ and $\tilde{m}^* = H_m(m^*, ID^*, upk_{ID^*})$. If $F(u^*) = 0 \bmod p$ and $K(\tilde{m}^*) = 0 \bmod p$, C_1 calculates a solution of the given CDH problem:

$$g^{ab} = \frac{\left(\sigma_1^*\right)^{\tau^*}}{\left(\sigma_2^*\right)^{J(u^*)} \left(\sigma_3^*\right)^{L(m^*)}} \tag{6.3}$$

Similar to Shim's scheme, the probability that C_1 successfully computes the CDH solution is

$$\varepsilon_1 \cdot \frac{1}{16 \cdot C_s \cdot (C_{par} + C_s) \cdot (n_u + 1) \cdot (n_m + 1)}. \tag{6.4}$$

If the CDH problem is intractable in polynomial time, our improved CLS scheme is resistant to MBPK attacks from Type II attackers.

7. Performance comparison

Table 2 shows the security analysis comparison with similar CLS schemes [11, 18, 19, 22] without random oracles, where the symbols ✓ and × represent the scheme's ability or inability to resist such attackers. Obviously, the improved CLS scheme can resist Type I and Type II attackers, making it more suitable for the IoT environment.

Table 2. Comparison of security with similar standard model CLS schemes.

Scheme	Type I attacker	Type II attacker
Scheme [11]	×	✓
Scheme [18]	✓	×
Scheme [19]	×	✓
Scheme [22]	×	×
Our scheme	✓	✓

In the **CL-Vf** algorithm of the improved scheme, $e(g, pk^{(2)})$, Z and $e(\tilde{U}, pk^{(1)})$ can be pre-computed since they are independent of the signed message. Hence, our enhanced scheme inherits the performance of Shim's scheme [11] for computing and communication. Table 3 shows the computational and communication costs obtained from the analysis of the Shim scheme [11] and our proposed improved scheme, where T_p and T_m represent the execution of bilinear mapping and point-scalar multiplication operations, and $|G_1|$ represents the byte length of elements in G_1.

Table 3. Comparison of communication and computational costs.

Scheme	CL-Sig	CL-Vf	Signature length
Scheme [11]	$5T_m$	$3T_p$	$3\|G_1\|$
Our scheme	$5T_m$	$3T_p$	$3\|G_1\|$

We used pbc 0.5.14 library [24] and A-type elliptic curve parameters for computations and evaluated the average execution time of cryptographic operations. The experimental environment was Ubuntu 22.04.2 LTS system with Intel(R) Xeon(R) Gold 6133 CPU @ 2.50GHz. The time required for bilinear pairing and point scalar multiplication was calculated to be 3.21 milliseconds and 1.15 milliseconds, respectively. Figure 2 shows the required running time for signers and verifiers, which is suitable for device time consumption in the IoT environment.

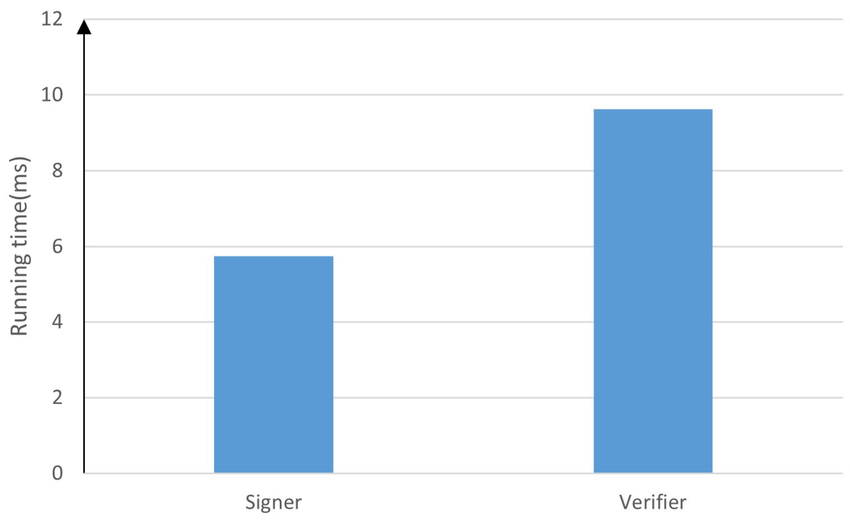

Figure 2. Computational overhead for the signer and the verifier.

8. Conclusions

To address the issues of data integrity verification and identity authentication in the IoT environment, we chose the CLS method. Shim [11] designed a CLS scheme without random oracles and demonstrated its security in the standard model. In this article, we provide an attack against Shim's scheme [11] and found their scheme to be vulnerable to PKR attacks. In addition, we proposed an improved scheme to fix the security vulnerabilities of their scheme and combined it with blockchain to further enhance security. Finally, the analysis results show that our enhanced scheme achieves stronger security while preserving the performance of the original scheme. Although bilinear pairing operations consume relatively more time than other operations, this improved scheme still involves bilinear pairing operations. We plan to reduce the number of bilinear pairing operations for higher operational efficiency while ensuring security in the future.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61662069), the China Postdoctoral Science Foundation (No. 2017M610817).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. P. Pradeep, K. Kant, Conflict detection and resolution in IoT systems: a survey, *IoT*, **3** (2022), 191–218. https://doi.org/10.3390/iot3010012
2. Y. Wu, H. N. Dai, H. Wang, Z. Xiong, S. Guo, A survey of intelligent network slicing management for industrial IoT: integrated approaches for smart transportation, smart energy, and smart factory, *IEEE Commun. Surv. Tutorials*, **24** (2022), 1175–1211. https://doi.org/10.1109/COMST.2022.3158270
3. I. Yoosefdoost, M. Basirifard, J. Álvarez-García, Reservoir operation management with new multi-objective (MOEPO) and metaheuristic (EPO) algorithms, *Water*, **14** (2022), 2329. https://doi.org/10.3390/w14152329
4. M. Ataei Nezhad, H. Barati, A. Barati, An authentication-based secure data aggregation method in Internet of Things, *J. Grid Comput.*, **20** (2022), 29. https://doi.org/10.1007/s10723-022-09619-w
5. V. Muthukumaran, Efficient digital signature scheme for Internet of Things, *Turk. J. Comput. Math. Educ.*, **12** (2021), 751–755. https://doi.org/10.17762/turcomat.v12i5.1480
6. H. Kashgarani, L. Kotthoff, Is algorithm selection worth it? Comparing selecting single algorithms and parallel execution, in *AAAI Workshop on Meta-Learning and MetaDL Challenge*, PMLR, (2021), 58–64.
7. B. C. Hu, D. S. Wong, Z. Zhang, X. Deng, Certificateless signature: a new security model and an improved generic construction, *Des. Codes Cryptogr.*, **42** (2007), 109–126. https://doi.org/10.1007/s10623-006-9022-9
8. D. Rajan, P. Eswaran, G. Srivastava, K. Ramana, C. Iwendi, Blockchain-based multi-layered federated extreme learning networks in connected vehicles, *Expert Syst.*, **2022** (2022), e13222. https://doi.org/10.1111/exsy.13222
9. S. Tanwar, N. Gupta, C. Iwendi, K. Kumar, M. Alenezi, Next generation IoT and blockchain integration, *J. Sens.*, **2022** (2022), 9077348. https://doi.org/10.1155/2022/9077348
10. R. Ch, D. J. Kumari, T. R. Gadekallu, C. Iwendi, Distributed-ledger-based blockchain technology for reliable electronic voting system with statistical analysis, *Electronics*, **11** (2022), 3308. https://doi.org/10.3390/electronics11203308
11. K. A. Shim, A new certificateless signature scheme provably secure in the standard model, *IEEE Syst. J.*, **13** (2018), 1421–1430. https://doi.org/10.1109/JSYST.2018.2844809
12. S. S. Al-Riyami, K. G. Paterson, Certificateless public key cryptography, *Asiacrypt*, **2894** (2003), 452–473.
13. S. Hussain, S. S. Ullah, I. Ali, J. Xie, V. N. Inukollu, Certificateless signature schemes in Industrial Internet of Things: A comparative survey, *Comput. Commun.*, **181** (2022), 116–131. https://doi.org/10.1016/j.comcom.2021.10.010
14. Y. Chen, D. Zheng, R. Guo, Y. Zhang, X. Tao, A blockchain-based revocable certificateless signature scheme for IoT device, *Int. J. Network Secur.*, **23** (2021), 1012–1027. https://doi.org/10.1109/TII.2021.3084753

15. S. Hussain, S. S. Ullah, A. Gumaei, M. Al-Rakhami, I. Ahmad, S. M. Arif, A novel efficient certificateless signature scheme for the prevention of content poisoning attack in named data networking-based internet of things, *IEEE Access*, **9** (2021), 40198–40215. https://doi.org/10.1109/ACCESS.2021.3063490

16. R. Canetti, O. Goldreich, S. Halevi, The random oracle methodology, revisited, *J. ACM*, **51** (2004), 557–594. https://doi.org/10.1145/1008731.1008734

17. C. Wu, H. Huang, K. Zhou, C. Xu, Cryptanalysis and improvement of a new certificateless signature scheme in the standard model, *China Commun.*, **18** (2021), 151–160. https://doi.org/10.23919/JCC.2021.01.013

18. J. K. Liu, M. H. Au, W. Susilo, Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model, in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, ACM, (2007). https://doi.org/10.1145/1229285.1266994

19. H. Xiong, Z. Qin, F. Li, An improved certificateless signature scheme secure in the standard model, *Fundam. Inform.*, **88** (2008), 193–206.

20. Y. Yuan, D. Li, L. Tian, H. Zhu, Certificateless signature scheme without random oracles, in *Advances in Information Security and Assurance: Third International Conference and Workshops*, Springer, (2009), 31–40. https://doi.org/10.1007/978-3-642-02617-1_4

21. Q. Xia, C. X. Xu, Y. Yu, Key replacement attack on two certificateless signature schemes without random oracles, *Key Eng. Mater.*, **439** (2010), 1606–1611. https://doi.org/10.4028/www.scientific.net/KEM.439-440.1606

22. Y. Yu, Y. Mu, G. Wang, Q. Xia, B. Yang, Improved certificateless signature scheme provably secure in the standard model, *IET Inf. Secur.*, **6** (2012), 102–110. https://doi.org/10.1049/iet-ifs.2011.0004

23. Y. Yuan, C. Wang, Certificateless signature scheme with security enhanced in the standard model, *Inf. Process. Lett.*, **114** (2014), 492–499. https://doi.org/10.1016/j.ipl.2014.04.004

24. B. Lynn, PBC library–The pairing-based cryptography library, 2007. Available from: http://crypto.stanford.edu/pbc/.