A Secure Certificateless Signature Scheme for Space-Based Internet of Things

Tongwei Liu, Wei Peng, Kai Zhu, and Baokang Zhao

College of Computer, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Baokang Zhao; bkzhao@nudt.edu.cn

Received 11 July 2022; Accepted 5 October 2022; Published 15 November 2022

Academic Editor: Hao Peng

Copyright © 2022 Tongwei Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Space-based Internet of things (S-IoT) can provide global services and connection capabilities. It has broad emerging application, including marine monitoring, forest monitoring, animal monitoring, disaster emergency response and other fields. However, owing to the openness of satellite communications, S-IoT is vulnerable to hijacking attacks, and malicious attackers can tamper with or forge transmitted messages. More seriously, due to limited S-IoT node resources, it is difficult to directly apply existing security solutions to terrestrial networks to the S-IoT. In this study, we propose CSP, a novel, secure, and efficient scheme based on certificateless signatures and bilinear pairings for S-IoT. CSP consists of six phases: system setup, partial private key settings, private key settings, public key settings, signing and verifying. In CSP, we especially design that part of the private key comes from the ground KGC and the other part is generated by the communication entity itself. We theoretically prove CSP is secure, and it is able to resist tampering or forgery attacks. Moreover, it can also ensure the authenticity, integrity, unforgeability, and non-repudiation of transmitted messages. We also conducted extensive experiments and compared CSP with the existing schemes. The experimental results demonstrate that CSP can significantly reduce the total scheme time consumption. Especially, it can reduce about 50–60% of the time in the signature verification phase.

1. Introduction

Space-based Internet of Things (S-IoT) has broad application prospects in the fields of disaster emergency, animal monitoring, air pollution monitoring, aerospace measurement and control, aviation and navigation, and mobile communications [1–6], as shown in Figure 1. At present, the IoT generally transmits information through terrestrial networks such as the Internet and mobile communication networks, which severely limits the application scope and makes it hard to achieve true interconnection of all things. Terrestrial networks mainly cover densely populated areas such as cities and towns. More than 70% of the Earth’s area and more than 3 billion people are not covered by terrestrial networks. S-IoT can cover sparsely populated areas such as oceans, polar regions, and deserts [7, 8]. In addition, S-IoT can also be applied in situations where terrestrial infrastructure has been damaged, such as reconstruction in disaster areas. According to the forecast of McKinsey, an American consulting company, the output value of S-IoT will reach 560 billion to 850 billion dollars within the next 5 years. It is expected that the number of machine-to-machine (M2M) and IoT networks connected to S-IoT will reach 5.96 million by 2025 [9].

Authentication is one of the fundamental issues for security [10]. However, owing to the openness of satellite communications, identity authentication between nodes and message integrity authentication face significant challenges. First, in S-IoT, a single satellite usually needs to provide data transmission services for massive ground nodes. The particularity of the environment makes the messages transmitted by satellites vulnerable to security threats such as eavesdropping, tampering, and forgery. Therefore, the communication security between satellite nodes is indispensable. Second, since the communication bandwidth of a satellite is typically narrow and satellite storage resources are also limited, the authentication scheme must be efficient. Third, in S-IoT, scores of ground nodes are distributed in the
wild and may even be carried by animals. The computing,
storage, and energy resources of ground nodes are severely
limited due to the constraints of weight, volume, and de-
ployment environment. These constraints require that the
operations of ground nodes must be simple enough to
operate [11]. Therefore, a secure and effective authentication
scheme is urgently needed to ensure the development of
S-IoT.

The main contribution of this study can be summarized
as follows:

(1) We proposed CSP, a novel, secure, and efficient
scheme based on certificateless signatures and bilinear
pairings for S-IoT. To set the complete private
key, a partial private key comes from the ground
KGC and the other part of the private key is gen-
erated by the communication entity itself. In this
way, CSP does not need a certificate authority and
solves the key escrow issues.

(2) We have proved CSP, which has strong security and
can effectively resist the attacks of external and in-
ternal adversaries. Moreover, CSP enables authen-
ticity, integrity, unforgeability, and nonrepudiation
of transmitted messages.

(3) We designed the CSP so that it only needs one hash
function. Compared with the previous schemes, our
scheme reduces the bilinear pairing operations in the
verification phase. Performance evaluation shows that
CSP can significantly reduce the total scheme time
consumption. Especially, it can reduce about 50-60%
of the time in the signature verification phase.

The organization of this study is summarized as follows.
In Section 2, we review the related work. The S-IoT archi-
tecture, some principles of cryptography, and security model
are described in Section 3. In Section 4, we elaborate on our
proposed scheme for CSP. We present the security analysis
of CSP in Section 5. In Section 6, we evaluate the perfor-
ance. We finally conclude the study and discuss the future
research direction in Section 7.

2. Related Work

The existing work on the certification of IoT can be roughly
divided into three types. The first method is the authenti-
cation based on the public key infrastructure (PKI) mech-
anism, the second method is the authentication based on the
identity-based cryptosystem (IBC), and the third method is
the authentication based on the certificateless public key
cryptography (CL-PKC).

In the traditional PKI, the certificate authority (CA)
complexly manages the public key and identity information
of all users and issues certificates to authenticated users.
However, the generation, storage, distribution, verification,
and revocation of certificates can be resource intensive.

Shamir [12] introduced IBC to try to solve the tedious
problem of certificate management. As shown in [13], in
addition to human users, computers and servers, smartphones, other mobile devices, and IoT devices also have their
own identities. The IBC uses the user’s identity information
directly as the public key. The user’s private key is generated
by the key generation center (KGC) using the master key and
the user’s identity information. There have also been some
studies [14, 15] on IBC in recent years. However, in an IBC
system, the KGC must be highly trusted because it can
encrypt and decrypt messages on behalf of any system user.
This creates a key escrow problem inherent in the IBC
system. Once KGC is invaded or breached, all users’ private
keys and identity information will be leaked. Then, the entire
system will be paralyzed.
To solve these problems, Al-Riyami et al. [16] proposed CL-PKC. They designed a novel certificateless signature (CLS) scheme, in which KGC only provides part of the user’s private key, and the other part is generated by the user, which can solve the key escrow problem. In addition, different from the traditional PKI system, users in the CL-PKC system do not need to be authenticated. Therefore, CLS do not need a certificate authority. CLS can be divided into two categories: one contains bilinear pairs and the other does not contain them.

Later, CLS technology has been greatly developed. In 1996, Bellare and Rogaway [17] proposed a random oracle model (ROM) to prove the security of the CLS scheme. This model can efficiently evaluate the scheme, but there are many loopholes. At the beginning of this century, Chowdary Gorantla and Saxena [18] claimed to have proposed a high-efficiency CLS scheme. However, in 2006, Cao et al. [19] proved that their scheme is not secure against key substitution attacks. Zhang et al. [20] designed a new certificateless signature scheme using bilinear pairings and used the ROM to achieve strict security proofs under the assumption of computing the Diffie–Hellman Problem (DHP), but their algorithm’s computational cost is significant.

In 2012, Tso et al. [21] proposed a certificateless short signature scheme. However, Du and Wen [22] point out that the scheme cannot resist the attack of the second type of strong adversary in the ROM model. There are also two short CLS schemes [23, 24], both of which have been proven to be secure. However, the scheme in [23] needs to perform two bilinear calculations during verification, while the scheme in [24] requires three bilinear calculations. Obviously, the computational cost is relatively high. The first CLS scheme without bilinear pairings is proposed in [25], but it is pointed out in [26] that this scheme is vulnerable to the second type of strong adversaries. After that, proposed CLS schemes without bilinear pairings were proposed in [27, 28], respectively. However, it is pointed out in [29, 30] that both schemes are vulnerable to the first type of super-adversaries. In 2015, Hassouna et al. [31] claimed to propose a strongly secure CLS scheme and proved its security under the assumption of two classes of strong adversary attacks. However, scholars soon proved that the scheme in [31] is insecure in the face of the attack of the first type of strong adversary. In addition, Wang et al. [32] proposed a novel, reliable, and efficient pairing-free certificateless scheme for the Industrial Internet of Things (IIoT) that utilizes the state-of-the-art blockchain technique and smart contracts. In 2018, Jia et al. [33] proposed an efficient and nonbilinear pairwise CLS scheme suitable for the IoT. In 2020, Du et al. [34] found that the scheme in [33] could not resist the attack of the first type of adversaries and proposed a new scheme. In the same year, a bilinear pair-free CLS scheme suitable for resource-constrained scenarios was proposed in [35]. However, in 2021, Xu et al. [36] found that the solution of [35] was vulnerable to signature forgery attacks and could not achieve its purpose.

Table 1 shows the main mechanisms and shortcomings of the existing works. From the above, we can see that the existing PKI and IBC mechanisms may not be suitable for S-IoT. The CLS scheme without bilinear pairings does not seem to be reassuring in terms of security. Especially, in recent years, the scheme proposed in a short period of time will be found to be unsafe. However, existing CLS schemes containing bilinear pairings have a large overhead. If these schemes are applied to the S-IoT, they will take up a large amount of resources. Therefore, we want to design a CLS scheme that balances security and computational overhead, which is suitable for the special environment of S-IoT.

3. Preliminaries

3.1. S-IoT Architecture. The typical S-IoT architecture [3] is shown in Figure 2. The S-IoT architecture consists of three parts: space segment, ground segment, and user terminal. The space segment consists of a constellation of satellites. The ground segment mainly includes the ground stations and the control stations. The user terminal refers to various terminals which are mainly used to send and receive signals. Information security is an important issue in S-IoT. There may be malicious nodes attacking the S-IoT system through eavesdropping, forgery, tampering, and other means.

3.2. Elliptic Curve. Elliptic curve cryptography (ECC) is a method of constructing cryptographic schemes from elliptic curves over finite fields. Elliptic curve ciphers can achieve the same strength with shorter keys than RSA; that is, elliptic curve ciphers have shorter key lengths but higher strength. In general, an elliptic curve cipher with a key length of 160 bits can achieve the same strength as RSA with a key length of 1024 bits.

Let $p$ be a large prime number of length $\lambda$, $\text{GF}(p)$ represents a finite field, and an elliptic curve is a series of points satisfying the following equations:

$$G = \{(x, y): y^2 = x^3 + ax + b, 4a^3 + 27b^2 \text{mod} p \neq 0\} \cup O$$

$$a, b \in \text{GF}(p),$$

where $O$ represents the point at infinity.

3.3. Bilinear Pairing. Let the bilinear mapping be $e: G_1 \times G_2 \rightarrow G_3$, where $G_1$ and $G_2$ are the additive cyclic group and the multiplicative cyclic group of order prime $p$, respectively. The generator of $G_1$ is $P$. The bilinear map satisfies the following properties:

(1) Bilinearity: $\forall Q, W, Z \in G_1$; there are $e(Q, W + Z) = e(Q, W) \cdot e(Q, Z)$ and $(Q + W, Z) = e(Q, Z) \cdot e(W, Z)$. Then, $\forall a, b \in Z_q^*$, $e(aQ, bW) = e(Q, W)^{ab}$

(2) Nondegenerate: $\exists P, Q \in G_1, e(P, Q) \neq 1$

(3) Computability: $\forall P, Q \in G_1$; there is a valid algorithm that can compute $e(P, Q)$

3.4. Elliptic Curve Computational Diffie–Hellman Problem (ECDHP). Given $aP, bP \in G_1, a, b, c \in Z_q^*$, where $P$ is the generator of $G_1$, it is hard to calculate $abP \in G_1$. 
3.5. Bilinear Diffie–Hellman Problem (BDHP). Given $aP, bP, cP \in G_1$, $a, b, c \in \mathbb{Z}_q^*$, where $P$ is the generator of $G_1$, it is hard to calculate $e(P, P)^{abc} \in G_2$.

3.6. Certificateless Signature. A certificateless signature scheme generally includes three entities: KGC, signer, and verifier. The general steps of the CLS are as follows:

1. Setup: this step is performed by KGC. We input a security parameter $l$ and output the system master key $s$ and public parameter $params$. KGC securely keeps the system master key $s$ and makes $params$ public.

2. Set partial private key: this step is performed by KGC. We input the system master key $s$, the public parameter $params$ and the signer’s identity $ID$. Then, we output the partial private key $D$.

3. Set private key: this step is performed by the signer. We input the public parameters $params$, the signer’s identity $ID$, the partial private key $D$, and the signer’s secret value. We output the private key $sk$.

4. Set public key: this step is performed by the signer. We input the public parameter $params$ and the signer’s secret value and output the public key $pk$.

5. Sign: this step is performed by the signer. We input public parameters $params$, message $m$, the signer’s $ID$, the private key $sk$, and the public key $pk$ and output the signature $\sigma$.

6. Verify: this step is performed by the verifier. We input public parameters $params$, message $m$, signer $ID$, the public key $pk$, and the signature $\sigma$ and, finally, output

\[\text{Verify}(m, \sigma, params, ID, pk) \rightarrow \text{true or false}.\]

3.7. Security Model of CLS. As mentioned in [37, 38], traditional security controls and detection systems are often tailored against external threats, but insider attacks are also an ever-increasing threat to a system with dire consequences. For comprehensive consideration of safety, in a certificateless cryptosystem, external adversary and internal adversary $A_1$ and $A_2$ can be assumed [12]. At first, there are normal and strong adversaries in ROM. In [39], the adversary is expanded and classified into normal, strong, and super three levels. Through oracle queries, a normal adversary can only obtain the valid signature of the entity with the original public key. If the entity’s public key is replaced, the normal adversary cannot obtain a valid signature. If the public key of a strong adversary has been replaced, the
adversary cannot obtain a valid signature until providing the associated secret value of the new public key. A super adversary can obtain the valid signature of the entity whose public key has been replaced without the new secret value. Here, we consider the case of super adversary attacks.

The first type of adversary \( A_1 \): these kinds of adversaries are also called external adversaries. The adversary can replace the public key of the target entity, but cannot obtain the master key of the KGC and partial private key of the entity.

The second type of adversary \( A_2 \): these kinds of adversaries are also called internal adversaries. The adversary knows the master key of KGC and the partial private key, but cannot replace the public key of the target entity.

### 3.7.2. Type-II Model

Setup: challenger \( C \) executes the algorithm to get the master secret key \( s \) and public parameters \( \text{params} \). Then, \( C \) keeps \( s \) as secret and makes the \( \text{params} \) public.

Queries: \( A_1 \) adaptively asks one of the following queries to \( C \).

1. Public key extraction query: \( A_1 \) obtains the public key \( pk_i \) of \( ID_i \)
2. Replace public key query: \( A_1 \) replaces \( pk_i \) with \( pk_i' \) which \( A_1 \) chooses
3. Private key extraction query: \( A_1 \) obtains the private key \( sk_i \) of \( ID_i \)
4. Signature query: \( A_1 \) obtains a valid signature \( \sigma \) for \( (ID_i, m) \)
5. Output: finally, \( A_1 \) outputs a valid forgery \( \sigma^* \) for \( (ID_i', m^*) \), where

\[
\text{Verify}(m^*, \sigma^*, \text{params}, ID_i', pk_i') \rightarrow \text{true}
\]

### 3.7.2. Type-II Model

Setup: challenger \( C \) executes the algorithm to get the master secret key \( s \) and public parameters \( \text{params} \). Then, \( C \) keeps \( s \) as secret and makes the \( \text{params} \) public.

Queries: \( A_2 \) adaptively asks one of the following queries to \( C \).

1. Public key extraction query: \( A_2 \) obtains the public key \( pk_i \) of \( ID_i \)
2. Partial private key extraction query: \( A_2 \) obtains the partial private key \( D_{i'} \) of \( ID_{i'} \)
3. Signature query: \( A_2 \) obtains a valid signature \( \sigma \) for \( (ID_i, m) \)
4. Output: finally, \( A_2 \) outputs a valid forgery \( \sigma^* \) for \( (ID_i', m^*) \), where

\[
\text{Verify}(m^*, \sigma^*, \text{params}, ID_i', pk_i') \rightarrow \text{true}
\]

### 4. Proposed CSP Scheme

Our proposed scheme, CSP, is as follows:

Setup (KGC): the function of KGC is completed by the network control center (NCC) on the ground. KGC inputs security parameter \( k \) and selects elliptic curve addition cyclic group \( G_1 \) and multiplication cyclic group \( G_2 \) with order \( q \). The generator of \( G_1 \) is \( P \). We set up bilinear mapping \( e: G_1 \times G_1 \rightarrow G_2 \). KGC selects a random number \( s \in Z_q^* \) as the system master key and calculates \( P_{\text{pub}} = s \cdot P \) as the system public key. KGC securely saves the master key \( s \) and makes the parameter \( \text{params} \) = \((q, G_1, G_2, e, P, P_{\text{pub}}, H_1)\) to the public.

Set partial private key (KGC): after KGC receives \( ID_i \) from the entity (the satellite or the user on the ground) \( S_i \), it calculates \( Q_i = H_1(ID_i) \) and then calculates the partial private key \( D_i = sQ_iP \) of \( S_i \).

Set public/private key (entity): KGC sends \( D_i \) to \( S_i \). \( S_i \) randomly selects the secret value \( x_i, x_i' \in Z_q^* \), and then calculates \( X_i = x_iP, Q_i = H_1(ID_i) \), and \( Z_i = x_i'P, Y_i = x_iP_{\text{pub}} \). We take \((X_i, Y_i)\) as the public key \( pk_i \) and take \((D_i, Z_i)\) as the private key \( sk_i \).

Sign (entity): when \( S_i \) needs to sign a message \( m \), the specific description is as follows:

1. \( S_i \) randomly selects a large integer \( a \in Z_q^* \)
2. \( S_i \) calculates \( M_i = x_i'^2H_1(m) \in Z_q^* \)
3. \( S_i \) calculates \( N_i = ax_iQ_i \in Z_q^* \)
4. \( S_i \) calculates \( s_i = e(M_iD_i, Z_i)^a \)
5. \( S_i \) sends \( s_i = (m, N_i, s_i) \) as a signature

Verify (entity): when another entity \( S_j \) receives the message \( m \) with the signature \( s_i \), it uses the public key \( pk_i \) of \( S_i \) to verify the signature. The specific description is as follows:

1. \( S_j \) calculates \( M_j = H_1(m) \in Z_q^* \)
2. \( S_j \) calculates \( e(M_jN_iX_iY_i) \).
3. \( S_j \) verifies that \( s_i \) = \( s_j \). If the equation holds to prove that the signature is valid; otherwise, the verification fails, the message is discarded, and a reauthentication message \( rm \) is returned.

Correctness analysis:

\[
\begin{align*}
  s_i &= e(M_iD_i, Z_i) \\
  &= e(P, P)^{sM_iQ_iX_i} \\
  &= e(P, P)^{sH_1(m)Q_iX_i} \\
  &= e(H_1(m)ax_iQ_i, X_{sx}P) \\
  &= e(M_jN_iX_iY_i) \\
  &= s_j.
\end{align*}
\]
As shown in Figure 3, the steps of interaction between satellite nodes are as follows:

Step 1 (KGC): KGC generates system parameters \( p = (q, G_1, G_2, e, P, P_{pub}, H_1) \) to the public. Then, KGC calculates partial private keys. KGC sends partial private keys to the corresponding satellites. In this scene, we take satellites \( S_i, S_j, \) and \( S_k \) as an example.

Step 2 \( (S_i, S_j, S_k) \): \( S_i \) randomly selects the secret value \( x_i, x_i' \in \mathbb{Z}_q^* \) and then calculates \( X_i, Q_i, Z_i, Y_i, X_i', Q_i', Z_i', Y_i'. \) We take \((X_i, Y_i)\) as the public key \( pk_i \). We take \((X_i', Y_i')\) as the public key \( pk_i' \). We take \((D_i, Z_i)\) as the private key \( sk_i \). We take \((D_i, Z_i')\) as the private key \( sk_i' \).

Step 3 \( (S_i) \): when \( S_i \) needs to sign a message \( m \), \( S_i \) generates \( \sigma_i = (m, N_i, s_i) \) as the signature of the message \( m \) to \( S_i \) and generates \( \sigma_i' = (m', N_i', s_i') \) as the signature of the message \( m' \) to \( S_k \).

Step 4 \( (S_j, S_k) \): when satellites \( S_j \) and \( S_k \) receive the message \( m \) and \( m' \) with the signature \( \sigma_i \) and \( \sigma_i' \), \( S_j \) uses the public key \( pk_i \) to verify the signature \( \sigma_i \) and \( S_k \) uses the public key \( pk_i' \) to verify the signature \( \sigma_i' \).

As shown in Figure 4, the steps of interaction between satellite and ground nodes are as follows:

Step 1 (KGC): KGC generates system parameters \( p = (q, G_1, G_2, e, P, P_{pub}, H_1) \) to the public. Then KGC calculates partial private keys. KGC sends partial private keys to the corresponding satellite and ground nodes. In this scene, take the satellite \( S_i \) and the ground node \( S_j \) as an example.

Step 2 \( (S_i, S_j) \): \( S_i \) randomly selects the secret value \( x_i, x_i' \in \mathbb{Z}_q^* \) and then calculates \( X_i, Q_i, Z_i, Y_i, X_i', Q_i', Z_i', Y_i'. \) We take \((X_i, Y_i)\) as the public key \( pk_i \) and take \((D_i, Z_i)\) as the private key \( sk_i \). Step 3 \( (S_i) \): \( S_i \) generates \( \sigma_i = (m, N_i, s_i) \) as the signature. Step 4 \( (S_i) \): when the ground node \( S_j \) receives the message \( m \) with the signature \( \sigma_i \), it uses the public key \( pk_i \) of \( S_i \) to verify the signature \( \sigma_i \).

5. Security Analysis

Lemma 1. Under the attack of the first type of super adversary \( A_1 \), it is assumed that \( A_1 \) can adaptively perform \( q_{H_1} \) for \( H_1 \), oracle queries, \( q_{sk} \) for private key extraction queries, \( q_{pk} \) for private key extraction queries, \( q_{sk} \) for private key extraction queries, \( q_{sk} \) for signature queries; there is an algorithm \( C \) that can solve the ECDHP problem with the advantage of \( \epsilon \geq \varepsilon (1/q_{H_1}) (1 - (1/q_{H_1}))^{\gamma_{pk}}. \)

Proof. Let \( A_1 \) be a super adversary, and we assume that the challenge for \( C \) is to know that \( Z_i = x_iP \) (which can be obtained in the private key query below), and \( aP, a \in \mathbb{Z}_q^* \), \( C \) calculates \( ax_i'P \) after interacting with \( A_1 \). We play the game as follows:

Game 1: challenger \( C \) inputs the security parameter \( l \), runs the system establishment algorithm to generate the system master key \( s \) and system parameter \( p \), then sends the params to \( A_1 \), and saves \( s \) in secret.

After going through all the queries, \( A_1 \) outputs a forged signature \( (m', N', s') \); if the forgery meets the following requirements, the super adversary \( A_1 \) is considered to win.

\[(1) \ A_1 \ \text{has never submitted} \ (ID^*, m^*) \ \text{to the signature oracle} \]

\[(2) \ A_1 \ \text{never submitted} \ ID^* \ \text{to partial private key oracles} \]

\[(3) \ \text{Verify} \ (m^*, a^*, \text{params}, ID^*, pk_i') \longrightarrow \text{true} \]

\( H_1 \) oracle query: \( C \) maintains a list \( L_{H_1} \) consisting of triples \((ID_1, Q_i, M_i)\), and the list is initially empty. When \( A_1 \) asks for \( ID_1 \) with identity \( ID_1 \), if \( ID_1 \) has been stored in \( L_{H_1} \), then \( C \) returns the corresponding value to \( A_1 \); otherwise, \( C \) calculates \( Q_i = nP, n \in \mathbb{Z}_q^* \); let \( M_i = r_1, r_1 \in Z_q^* \); we insert the new tuple \((ID_1, Q_i, M_i)\) into the list \( L_{H_1} \) and return to \( A_1 \).

Setup: \( C \) runs the system algorithm, selects a generator \( P \), and calculates \( P_{pub} = sP \), where \( s \) is the system master key that \( C \) does not know; in this game, \( C \) randomly selects an identity \( ID^* \), generates system parameters \( p = (P, P_{pub}, H_1) \), and send to \( A_1 \).

Public key extraction query: \( C \) maintains a list \( L_{pk} \) consisting of triples \((ID_1, x_i, pk_i)\), and the list is initially empty. When \( A_1 \) inputs \( ID_1 \) to ask, if \( ID_1 \) has been stored in \( L_{pk} \), \( C \) returns the corresponding value to \( A_1 \); otherwise, \( C \) calculates \( X_i = x_iP, x_i \in Z_q^* \) and returns the value to \( pk_i \). Then, \( C \) inserts \((ID_1, x_i, pk_i)\) into the list \( L_{pk} \).

Public key query: when \( A_1 \) enters \((ID_1, pk_i)\) query, if the tuple \((ID_1, x_i, pk_i)\) corresponding to \( ID_1 \) exists in the list \( L_{pk} \), \( C \) sets \( pk_i = pk_i' \) and sets \((ID_1, x_i, pk_i')\) and returns the list \( L_{pk} \); otherwise, \( C \) performs the public key generation step to generate \((ID_1, x_i, pk_i)\), then sets \( pk_i = pk_i' \), and returns \((ID_1, x_i, pk_i')\) to the list \( L_{pk} \).

Private key extraction query: \( C \) maintains a list \( L_{sk} \) consisting of four tuples \((ID_1, x, sk, Z_1)\), and the list is initially empty. When \( A_1 \) asks with \( ID_1 \), if \( ID_1 \) has been stored in \( L_{sk} \), \( C \) returns the corresponding value to \( A_1 \); otherwise, \( C \) calculates \( Z = xP, x \in Z_q^* \) and converts the new tuple \((ID_1, x, sk, Z_1)\) which is inserted into the list \( L_{sk} \) and returned to \( A_1 \).

Signature query: when \( C \) receives a \((ID_1, m_1)\) signature query, it performs the following steps:

\[(1) \ \text{If} \ ID = ID^*, \ C \ \text{aborts the query and outputs an error}; \ \text{otherwise}, \ C \ \text{queries} \ (ID_1, x_i, pk_i, (ID_1, x_i, pk_i')) \ \text{from} \ L_{pk}, L_{sk}, L_{H_1} \]

\[(2) \ \text{Calculate} \ N_i = ax_iQ_i, a \in Z_q^* \]

\[(3) \ \text{Calculate} \ s_1 = e(M_1D_1, Z_1)^{s_i} \]

\( A_1 \) aborts the query and outputs the signature \( \sigma = (N', s') \) of the identity \( ID^* \) on the message \( m^* \), which satisfies the verification condition:
with an nonnegligible probability: 

It can be known that \( V' = ax'_P \), which solves the ECDHP problem. Set events \( E_1, E_2 \) and \( E_3 \) are as follows:

- \( E_1: A_1 \) goes through a series of queries and \( C \) does not abort.
- \( E_2: A_1 \) successfully forges a valid signature.
- \( E_3: \) there is \( ID = ID^* \) in forged signature.

We set \( \Pr[E_2] = \varepsilon \), then obviously we have:

\[
\Pr[E_1] \geq \left( 1 - \frac{1}{q_H} \right)^{\varepsilon / q_H},
\]

\[
\Pr[E_2] \geq \varepsilon,
\]

\[
\Pr[E_3 \mid E_1, E_2] \geq \frac{1}{q_H}.
\]

It can be calculated that \( C \) can solve the ECDHP problem with a nonnegligible probability: \( \varepsilon' \geq \varepsilon / q_H (1 - (1/q_H))^{\varepsilon / q_H} \).

Lemma 2. Under the attack of the second type of super adversary \( A_2 \), it is assumed that \( A_2 \) can adaptively perform \( q_H \) for \( H_1 \) oracle queries, \( q_d \) for partial private key extraction queries, \( q_{sk} \) for private key extraction queries, \( q_{pk} \) for public key extraction queries, and \( q_d \) for signature extraction queries; there is an algorithm \( C \) that can solve the BDHP problem with the advantage of \( \varepsilon' \geq \varepsilon / q_H (1 - (1/q_H))^{\varepsilon / q_H} \).

Proof. Let \( A_2 \) be a super adversary. We assume that the challenge for \( C \) is that given the master key \( s \), \( D_l = sQ_P \) (which can be obtained in the partial private key query below), \( b = sQ_s, aP \) and \( cP, a, c \in Z_q^* \) an calculate \( S^* = e(P, P)^{abc} \) after interacting with \( A_2 \).

Game 2: challenger \( C \) inputs the security parameter \( l \), runs the system establishment algorithm to generate the system master key \( s \) and system parameter \( \text{params} \), then sends the params to \( A_2 \), and saves \( s \) in secret.

After going through all the queries, \( A_2 \) outputs a forged signature \((m^*, N^*, S^*)\); if the forgery meets the following requirements, the super adversary \( A_2 \) is considered to win.

(1) \( A_2 \) has never submitted \((ID^*, m^*)\) to the signature oracle

(2) \( A_2 \) never submitted \( ID^* \) to private key oracles

(3) Verify \((m^*, \sigma^*, \text{params}, ID^*, pki) \rightarrow \text{true}\)

\( H_1 \) oracle query: \( C \) maintains a list \( L_{H_1} \) consisting of triples \((ID_i, Q_i, M_i)\), and the list is initially empty. When \( A_2 \) asks \( C \) for \( H_1 \) with identity \( ID_i \), if \( ID_i \) has been stored in \( L_{H_1} \), then \( C \) returns the corresponding value to \( A_2 \); otherwise, \( C \) calculates \( Q_i = nP, n \in Z_q^* \); let \( M_i = r_j, r_j \in Z_q^* \), and we insert the new tuple \((ID_i, Q_i, M_i)\) into the list \( L_{H_1} \), and return to \( A_2 \).

Setup: \( C \) runs the system algorithm, selects a generator \( P \), and calculates \( P_{pub} = sP \), where \( s \) is the system master key that \( C \) does not know; in this game, \( C \) randomly selects an identity \( ID^* \), generates system parameters \( \text{params} = (P, P_{pub}, H_1) \), and sends to \( A_2 \).

Public key extraction query: \( C \) maintains a list \( L_{pk} \) consisting of triples \((ID_i, x_i, pki)\), and the list is initially empty. When \( A_2 \) inputs \( ID_i \) to ask, if \( ID_i \) has been stored in \( L_{pk} \), \( C \) returns the corresponding value to \( A_2 \); otherwise, \( C \) calculates \( x_i = xP, y = xP, x \in Z_q^* \) and returns the value to \( pki \). Then, \( C \) inserts \((ID_i, x_i, pki)\) into the list \( L_{pk} \).

Private key extraction query: \( C \) maintains a list \( L_{D_2} \), consisting of triples \((ID_i, Q_i, D_i)\), and the list is initially empty. When \( A_2 \) asks \( C \) with identity \( ID_i \), if \( ID_i = ID^* \), \( C \) aborts and outputs an error; otherwise, if \( ID_i \) has been stored in \( L_{D_2} \), \( C \) returns the corresponding value to \( A_2 \); if \( ID_i \) is not stored in \( L_{D_2} \), then \( C \) extracts the tuple \((ID_i, Q_i, M_i)\) from the list \( L_{H_1} \), calculates \( D_i = sQ_P \), and returns it to \( A_2 \).

Signature query: when \( C \) receives a \((ID_i, m_i)\) signature query, it performs the following steps:

(1) If \( ID = ID^* \), \( C \) aborts the query and outputs an error; otherwise, \( C \) queries \((ID_i, Q_i, M_i)\), \((ID_i, x_i, pki)\), \((ID_i, Q_i, M_i)\) from \( L_{D_2}, L_{pk}, L_{H_1} \)

(2) We calculate \( N_i = axQ_s, a \in Z_q^* \)

(3) We calculate \( s_i = e(MID_i, Z_i) \) and sign the message using \((N_i, s_i)\). \( C \) sends \((N_i, s_i)\) to \( A_2 \)

\( A_2 \) aborts the query and outputs the signature \( \sigma = (N_i, S_i) \) of the identity \( ID^* \) on the message \( m^* \), which satisfies the verification condition:

\[
S_i = e(V, V')
\]

\[
= s_i
\]

\[
= e(MID_i, Z_i)
\]

\[
= e(P, P)^{abc}.
\]

The premise that the signature satisfies the verification conditions is the parameter \( c = N_i x_i^2 \) which is known to \( A_2 \). Thus, \( A_2 \) can calculate \( S^* = e(P, P)^{abc} \) which solves the BDHP problem. We set events \( E_1, E_2 \), and \( E_3 \) are as follows:
\( E_1: A_2 \) goes through a series of queries and \( C \) does not abort.
\( E_2: A_2 \) successfully forges a valid signature.
\( E_3: \) there is \( ID = ID^* \) in forged signature.

We set \( \Pr[E_2|E_1] \geq \varepsilon \); then, obviously we have
\[
\Pr[E_1] \geq \left( 1 - \frac{1}{q_H} \right)^{\delta_\tau q_H},
\]
\[
\Pr[E_2|E_1] \geq \varepsilon,
\]
\[
\Pr[E_3|E_1,E_2] \geq \frac{1}{q_H}.
\]

It can be calculated that \( C \) can solve the BDHP problem with a nonnegligible probability:
\[
\varepsilon \geq \varepsilon/\sqrt{H} \left( 1 - (1/q_H) \right)^{\delta_\tau q_H}.\]
This proof is unforgeable against adaptive selective messaging attacks on the signature. Therefore, it is proved that the scheme can guarantee the authenticity and integrity of the message under the attack of the second type of super adversary \( A_2 \).

The above is a formal analysis in the ROM, which can ensure the strong security of CSP. The informal analysis is as follows.

- **Authenticity**: CSP can realize the authentication of the message source and the authentication of the communication entity. This is determined by adding the identity information of the communication entity in the signature.

- **Integrity**: CSP can guarantee that data have not been tampered with or damaged. This is determined by adding a hash function to the message in the signature. Once the message is changed, the corresponding hash function will change, which will result in authentication failure.

- **Unforgeability**: CSP means that nobody except the communication entity itself can forge the signature. This is determined by the private key generation method of CSP. Only the ground KGC knows the secret value that generates the partial private key and only the communication entity itself knows the complete private key.

- **Nonrepudiation**: CSP requires that neither the sender nor the receiver can deny the transmission. This is determined by adding the identity information of the sender in the signature. Besides, the receiver must reply with a message indicating whether the verification was successful.

### 6. Performance Evaluation

In this section, we test the performance of CSP. We compare CSP with existing representative CLS schemes. To ensure a benchmark for comparison, CSP uses widely accepted parameters. The program runs on a virtual machine with an Intel(R) Core(TM) i7-9750H-CPU@2.60 GHZ and 16 GB of RAM, using the Ubuntu18.04LT operating system. Using the Type-A type in the PBC library, its security level is comparable to that of 1024 bit RSA. The Type-A of this library is constructed on the elliptic curve \( y^2 = x^3 + x \) in the finite field \( GF(p) \), where \( p \) is a large prime number of 160 bits. Assuming the message length is 128 bits.

Table 2 shows the time required for various operations in the simulation environment of this study. We can see that the bilinear pairing operation takes the longest, followed by the hash operation and the point multiplication operation. Since the scalar addition and multiplication operations take negligible time compared to other operations, we ignore the overhead of these two types of operations in the comparison. The value in the table is the average time of each operation 100 times.

Table 3 shows the efficiency comparison of CLS schemes for satellite-to-satellite nodes. In S-IoT, we first consider the case of authentication between satellite nodes. Since each satellite node provides services for massive ground nodes, we believe that the authentication of satellite nodes requires high security. Therefore, we consider that when a satellite wants to authenticate with other satellites, it needs to use different public and private keys. It means that the satellite node \( S_j \) uses \((s_{kj}, p_{kj})\) for authentication with the satellite node \( S_j \), while \( S_j \) uses \((s_{kj}, p_{kj})\) for authentication with the satellite node \( S_k (j \neq k) \). In S-IoT, the roles of users and signers who generate public and private keys can only be assumed by the satellite itself, which cannot be delegated to a third party. Therefore, when calculating the overhead of the signing in this case, the overhead of generating public and private keys is included. Due to the difference in algorithms between CLS schemes with bilinear pairings and those without, the CLS schemes without bilinear pairings often include the point addition operation and the modular inverse operation. However, the time of these two operations is relatively short. For a more intuitive comparison, these two operations are ignored here.

The values of security against \( A_1 \) and \( A_2 \) are determined according to which level of adversary attack (normal, strong, and super) the corresponding scheme can resist in ROM. Among these schemes, it is mentioned in [36] that the scheme in [35] was vulnerable to signature forgery attacks and was insecure against \( A_1 \) and the scheme in [31] has recently been shown to be insecure in the face of the attack of \( A_1 \).

It can be seen from Figure 5 that, in the simulation environment of this study, except for the unsafe schemes in the table, the scheme in [34] has the shortest time of 8.727 ms when signing. CSP needs 10.638 ms for signing, which is in the middle level of these schemes. However, in the verification and total time comparison, the overhead of CSP is the smallest, which are 4.681 ms and 15.319 ms. Compared with the scheme in [34], CSP improves the efficiency by about 50% and 15%, respectively. In the CLS schemes with bilinear pairs, CSP has the smallest overhead in both the signature and verification phases, which improves efficiency by about 9% and 60%, respectively. Besides, CSP can prove a higher level of security. The reason is that the scheme in [31] does not fully use the user’s public key when verifying the signature. It only uses part of the public key, which leads to additional verification if the public key is authentic and valid. The authenticity and validity of the public key will increase the overhead of the bilinear pairing operations.

Table 4 shows the efficiency comparison of CLS schemes for satellite-to-ground nodes. When the satellite node is authenticated with the ground node in S-IoT, it is troublesome and resource-consuming to frequently update the
Generate secret key

Calculate $D_i$, $Z_i$

KGC: This role is filled by the satellite manufacturer or operating company

Scheme Sign Verify Type Security against $A_1$ Security against $A_2$ Signature size

| Scheme                  | Sign            | Verify            | Type                          | Security against $A_1$ | Security against $A_2$ | Signature size |
|-------------------------|-----------------|-------------------|-------------------------------|------------------------|------------------------|----------------|
| Hassouna's et al. [31]  | $4T_{pm} + 3T_h + 1T_p + 3T_r$ | $4T_p + 1T_h$     | With bilinear pairs           | Insecure               | Strong                 | $|G_1| + |Z_q^*$ |
| CSP                     | $5T_{pm} + 2T_h + 1T_p + 3T_r$ | $1T_p + 1T_{pm} + 1T_h$ | With bilinear pairs           | Super                  | Super                  | $|G_1| + |Z_q^*$ |

**Table 3:** Comparison of CLS schemes for satellite-to-satellite nodes.

**Figure 3:** Interaction between satellite nodes.

**Table 4:** Comparison of CLS schemes for satellite-to-ground nodes.

| Scheme                  | Sign            | Verify            | Type                          | Security against $A_1$ | Security against $A_2$ | Signature size |
|-------------------------|-----------------|-------------------|-------------------------------|------------------------|------------------------|----------------|
| Hassouna's et al. [31]  | $1T_{pm} + 1T_h + 1T_p$ | $4T_p + 1T_h$     | With bilinear pairings        | Insecure               | Strong                 | $|G_1| + |G_2|$ |
| CSP                     | $1T_{pm} + 1T_h + 1T_p + 1T_{pm} + 1T_h$ | $1T_p + 1T_{pm} + 1T_h$ | With bilinear pairings        | Super                  | Super                  | $|G_1| + |Z_q^*$ |

Select $H_1 : \{0,1\}^* \rightarrow Z_q$

* and the master key $s$

Calculate $P_{pub} = s \cdot P$

Generate system parameters $params = (q, G_1, G_2, e, P, P_{pub}, H_1)$ and partial secret key $D = sQ, Q = H_1(ID)$

**Table 2:** Symbol description and operation time.
public and private keys. Therefore, when the satellite $S_i$ is authenticated with the ground node, it only uses $(sk_i, pk_i)$.

We can see from Figure 6 that the scheme in [34] has the shortest times of 0.886 ms in the signing phase, except for the unsafe scheme. CSP needs 4.791 ms, which is similar to the schemes in [31, 36]. In the verification phase and total time comparison, CSP still has the best performance, which improves efficiency by about 50% and 7%, respectively.

Finally, we discuss the issue of CSP complexity. For the scenario in Figure 5, to complete the authentication between satellite nodes, three authentication messages need to be transmitted. The communication cost of each authentication message is 480 bits. The signature cost is 10.638 ms, and the verification cost is 4.681 ms. For the scenario in Figure 6, three authentication messages between the satellite and the ground node need to be transmitted. The communication cost of each authentication message is 480 bits. The signature cost is 4.681 ms. The CSP performance is shown in Table 5.

**Figure 4:** Interaction between the satellite and ground nodes.

**Figure 5:** Comparison of CLS schemes for interaction between satellite nodes.
7. Conclusions and Future Work

Due to the special environment and restricted resources, the security solution used by the terrestrial network for communication cannot be directly applied to the S-IoT. This study proposes a strong, secure certificateless signature scheme with bilinear pairings named CSP, which is suitable for S-IoT. Before the satellite is connected to the space-based network, the manufacturer or the company that undertakes the work of KGC inputs the partial private key and public parameters to the satellite. The satellite uses its own identity information and secret value to calculate the public key, the private key, and the signature. CSP can ensure the authenticity, integrity, unforgeability, and nonrepudiation of transmitted messages. The CSP solves the problems of complicated certificate management in the traditional PKI system key escrow in the IBC system. In the future, the secure access of satellites facing a large number of ground nodes is an important issue. In recent years, an ultra-supercost authentication protocol for electric vehicles, charging by utilizing the characteristics of extended chaotic maps, has been proposed in [40] which can resist man-in-the-middle attacks, replay attacks, and impersonation attacks. This work provides us with new ideas on how to perform rapid authentication when considering a large number of nodes that want to access satellites. Besides, analysis shows that compared with similar schemes, CSP can achieve a higher security level and lower overhead. However, CSP still needs a relatively long time in the signature phase, which is the research direction of future work.

Data Availability

The data that support the findings of this study can be obtained from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (Grant no. 61972412).

References

[1] D. Yang, Y. Zhou, W. Huang, and X. Zhou, “5G mobile communication convergence protocol architecture and key technologies in satellite internet of things system,” *Alexandria Engineering Journal*, vol. 60, pp. 465–476, 2021.

[2] M. Bacco, L. Boero, P. Cassarà et al., “IoT applications and services in space information networks,” *IEEE Wireless Communications*, vol. 26, pp. 31–37, 2019.
Identity-Based Cryptosystems and Signature

[11] C. Fei, B. Jiang, K. Xu, L. Wang, and B. Zhao, “An intelligent cloud-linked ambient air quality monitoring apparatus for gaseous pollutants in urban areas,” Journal of Internet Services and Information Security (JISIS), vol. 11, no. 1, pp. 64–79, 2021.

[12] D. Palma and R. Birkeland, “Enabling the internet of arctic things with freely-drifting small-satellite swarms,” IEEE Access, vol. 6, pp. 71435–71443, 2018.

[13] T. Wei, W. Feng, Y. Chen, C. Wang, N. Ge, and J. Lu, “Hybrid Satellite-Terrestrial Communication Networks for the Maritime Internet of Things: Key Technologies, Opportunities, and Challenges,” 2019, http://www.w3.org/1999/xlink:href 1903.11814.

[14] P. Gung and P. Li, “Further improvement of a certificateless signature scheme without bilinear pairings,” International Journal of Communication Systems, vol. 25, no. 11, pp. 1432–1442, 2012.

[15] M. Tian and L. Huang, “Cryptanalysis of a certificateless signature scheme without pairings,” International Journal of Communication Systems, vol. 26, no. 11, pp. 1375–1381, 2013.

[16] P. Gong and P. Li, “Further improvement of a certificateless signature scheme without pairing,” International Journal of Communication Systems, vol. 27, no. 10, pp. 2083–2091, 2014.

[17] L. Wang, K. Chen, Y. Long, X. Mao, and H. Wang, “A modified efficient certificateless signature scheme without bilinear pairings,” in Proceedings of the Paper presented at the 2015 International Conference on Intelligent Networking and Collaborative Systems, Taipei, Taiwan, September 2015.

[18] K. H. Yeh, K. Y. Tsai, and C. Y. Fan, “An efficient certificateless signature scheme without bilinear pairings,” Multimedia Tools and Applications, vol. 74, no. 16, pp. 6519–6530, 2015.

[19] K. H. Yeh, C. Su, K. K. R. Choo, and W. Chiu, “A novel certificateless signature scheme for smart objects in the internet-of-things,” Sensors, vol. 17, 2018.

[20] M. Hassouna, E. B. M. Bashier, and B. I. A. Barry, “A strongly secure certificateless digital signature scheme in the random oracle model,” International Journal on Network Security, vol. 18, pp. 938–945, 2016.

[21] W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, “Blockchain-based reliable and efficient certificateless signature for IIoT devices,” IEEE Transactions on Industrial Informatics, vol. 18, no. 10, pp. 7059–7067, 2022.

[22] X. Jia, D. He, Q. Liu, and K. K. R Choo, “An efficient provably-secure certificateless signature scheme for internet-of-things deployment,” Ad Hoc Networks, vol. 71, pp. 78–87, 2018.

[23] H. Du, Q. Wen, S. Zhang, and M. Gao, “A new provably secure certificateless signature scheme for internet of things,” Ad Hoc Networks, vol. 100, Article ID 102074, 2020.

[24] G. Thumbrum, G. S. Rao, P. V. Reddy, N. B. Gayathri, and D. V. R. K. Reddy, “Efficient pairing-free certificateless signature scheme for secure communication in resource-constrained devices,” IEEE Communications Letters, vol. 24, no. 8, pp. 1641–1645, 2020.

[25] Z. Xu, M. Luo, M. K. Khan, K. K. R. Choo, and D. He, “Analysis and improvement of a certificateless signature scheme for resource-constrained scenarios,” IEEE Communications Letters, vol. 25, no. 4, pp. 1074–1078, 2021.
[37] A. Wall and I. Agrafiotis, “A Bayesian approach to insider threat detection,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 12, no. 2, pp. 48–84, 2021.

[38] F. L. Greitzer, J. Purl, and P. J. Sticha, “Use of expert judgments to inform bayesian models of insider threat risk,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 12, no. 2, pp. 3–47, 2021.

[39] X. Huang, Y. Mu, W. Susilo, D. S. Wong, and W. Wu, “Certificateless signatures: new schemes and security models,” *The Computer Journal*, vol. 55, no. 4, pp. 457–474, 2011.

[40] W. Wang, Z. Han, M. Alazab, T. R. Gadekallu, X. Zhou, and C. Su, “Ultra super fast authentication protocol for electric vehicle charging using extended chaotic maps,” *IEEE Transactions on Industry Applications*, vol. 58, no. 5, pp. 5616–5623, 2022.