IBE-Signal: Reshaping Signal into a MITM-Attack-Resistant Protocol

Shiqi Liu, Yan Shao, Hanbo Luo, and Hong Di

1School of Cyber Science and Engineering, University of International Relations, Beijing 100091, China
2School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

Correspondence should be addressed to Hong Di; di_hong@163.com

Received 18 November 2021; Revised 13 April 2022; Accepted 27 May 2022; Published 21 July 2022

The Signal Protocol is one of the most popular privacy protocols today for protecting Internet chats and supports end-to-end encryption. Nevertheless, despite its many advantages, the Signal Protocol is not resistant to Man-In-The-Middle (MITM) attacks because a malicious server can distribute the forged identity-based public keys during the user registration phase. To address this problem, we proposed the IBE-Signal scheme that replaced the Extended Triple Diffie–Hellman (X3DH) key agreement protocol with enhanced Identity-Based Encryption (IBE). Specifically, the adoption of verifiable parameter initialization ensures the authenticity of system parameters. At the same time, the Identity-Based Signature (IBS) enables our scheme to support mutual authentication. Moreover, we proposed a distributed key generation mechanism that served as a risk decentralization to mitigate IBE’s key escrow problem. Besides, the proposed revocable IBE scheme is used for the revocation problem. Notably, the IND-CPA security of the IBE-Signal scheme is proven under the random oracle model. Compared with the existing schemes, our scheme provided new security features of mutual authentication, perfect forward secrecy, post-compromise security, and key revocation. Experiments showed that the computational overhead is lower than that of other schemes when the Cloud Privacy Centers (CPCs) number is less than 8.

1. Introduction

Revelations of mass surveillance of communications have made consumers more privacy-aware. Scientists and developers have proposed security techniques for end-users even if they do not trust the service providers fully [1]. The Signal Protocol is a cryptographic protocol that secures the text messages of billions of people. It provides end-to-end encryption and is applied by secure communication tools such as WhatsApp, Facebook Messenger, and Microsoft Skype [2, 3]. The information of applications with the Signal Protocol and their monthly active users are shown in Table 1.

The Signal Protocol consists of two main sub-algorithms, X3DH and Double Ratchet. X3DH generates many sets of ephemeral key pairs for each user in addition to the permanent keys. Then a shared key is created for users by combining ephemeral and permanent key pairs. The Double Ratchet algorithm contains two ratchets [4], the Diffie–Hellman ratchet and the Symmetric-key ratchet. The Symmetric-key ratchet uses the key derivation function (KDF) chain to derive new keys. It offers a previous-key-based key to encrypt each message and attempt to provide Perfect Forward Secrecy (PFS). Even if an attacker cracks the key of one message, the keys of previous messages cannot be derived inversely. The included Diffie–Hellman ratchet aims at ensuring Post-compromise Security (PCS). Therefore, the Double Ratchet algorithm satisfies both PFS and PCS. Neither the keys generated before the key $K$ nor after can be calculated when $K$ is compromised.

Although having PFS and PCS, Signal Protocol cannot resist MITM attacks due to X3DH’s key distribution problem. Since there is no public key authentication in X3DH, the server can be an attacker against the system itself or a collaborator of the adversary.
The developers did consider the shortcomings of Signal Protocol to protect against MITM attacks and made some refinements. So each chat has a unique safety number [9] that enables one to verify the security of messages and specific contacts. As shown in Figure 1, the safety number is a hash of identity generated locally by both communicators, usually in a QR code. If a safety number has been marked as verified, any change can be manually approved before sending a new message [9]. However, a better plan to prevent MITM attacks should be in advance, rather than doing detection when an attack has approached. It is unreasonable to believe that a server will duly alert when it is a potential attacker. Only the user actively verifies the safety number through a third-party channel is reliable. Schröder et al. [10] found that most users could not successfully verify each other's safety numbers due to usability problems and incomplete mental models. Worsely, the Signal Protocol is not secure under a threat model with an untrusted server since much of the Signal Protocol’s functionality is on the server trusted premise.

The existence of MITM attacks is essentially a lack of public-key authentication. We are considering replacing X3DH with Identity-Based Encryption (IBE) for improvement. IBE scheme is a public-key cryptosystem where any string representing the identity is a valid public key [11]. A user can generate a public key from a known unique identifier, such as a phone number. Then a trusted third-party server calculates the corresponding private key from the public key. This process eliminates the need to distribute the public key before exchanging encrypted data. The sender can generate the public key and encrypt the data simply by using the receiver’s unique identifier. Correspondingly, the receiver can generate the private key with the help of a trusted third-party server, the Private Key Generator (PKG).

Blazy et al. [12] introduced IBS into the Signal Protocol to make it resistant to MITM attacks, but the problems [13–16] inherited in IBS/IBE are yet to be solved:

**The Authenticity of System Parameters Problem.** In the BF-IBE scheme, the attacker can generate the public system parameters by himself to make the users mistakenly believe that the parameters are correct.

**One-Way Authentication Problem.** Instant messaging protocols generally support mutual authentication, but BF-IBE only authenticates the recipient’s identity.

**Key Escrow Problem.** The PKG generates the users’ private keys with the master key. For this reason, identity theft might occur since the PKG is available to sign any message with the users’ identities. Users may also slander the PKG for misusing their private keys for signing, which threatens the undeniability of the signature.

**Public Key Revocation Problem.** A user can generate a new public-private key pair when his key is at risk or expire. However, the BF-IBE scheme is unavailable to revoke the user’s public key.

1.1. **Our Contributions.** Our advanced Signal Protocol has the following security features:

**Resist MITM attack.** Our IBE-Signal scheme is resistant to MITM attacks by leveraging the IBE scheme. Furthermore, we fixed the problems inherited in IBS/IBE.

**The Authenticity of System Parameters.** By introducing a verifiable parameter initialization technique, the public validation of parameters is guaranteed to ensure the correctness of parameters transmitted by the Key Generation Center (KGC) and Cloud Privacy Centers (CPCs).

**Mutual Authentication.** Our IBE-Signal scheme supports mutual authentication by introducing an Identity-Based Signature (IBS).

**Secure Key Escrow.** We used distributed key generation to solve the key escrow problem, inspired by the ESKI-IBE scheme proposed by Kumar et al. [17]. Moreover, we also used the KGC and the CPCs instead of one PKG. Compared with [17], we reduced the computational overhead since there is no need to negotiate parameters after an initial session.

**Resist User Slander.** The KGC and the CPCs own the master key share separately, which disenables an attacker from obtaining the complete key. Therefore, an attacker cannot decrypt the users’ ciphertexts or forge the users’ signatures, and users cannot defame the server for misusing their private key.

**Secure Key Issuing.** The blind signature enables the KGC or the CPCs to issue the private key without seeing the actual information sent by the user.

**Key Revocation.** We achieved efficient public key revocation by adopting Revocable IBE. This scheme proposed by Boldyreva et al. [18] is built on the Fuzzy IBE scheme and binary tree data structure.

**Perfect Forward Secrecy and Post-compromise Security.** Our scheme still retains the PFS and PCS of the Signal Protocol by introducing the concepts of session and connection in SSL protocol into our scheme and integrating the Double Ratchet algorithm into the CONNECTION part.
We proved the IND-ID-CPA security of the IBE-Signal scheme under the random oracle model and the Bilinear Diffie–Hellman (BDH) assumption. With more powerful attackers, we gained a more compact reduction than [17] in a challenger-adversary game.


The Signal Protocol cannot resist MITM attacks when the key registration is insecure. There are many ways to resist MITM attacks. The traditional methods are Digital Signatures and Message Authentication Codes. Rivest and Shamir [24] proposed the interlock protocol to expose the eavesdropper of full-duplex communication. However, the interlock protocol increases the communication cost and is not suitable for half-duplex communication. Khader and Lai [25] proposed a method to resist MITM attacks in the Diffie–Hellman Key Exchange Protocol with the Geffe generator.

The MITM attacks are due to the lack of public-key authentication, traditionally implemented by distributing public key certificates through the Public Key Infrastructure (PKI). In 1984, Shamir [15] proposed the IBE scheme to simplify the public key certificate management in the PKI. Boneh and Franklin [16] implemented a practical and functional IBE scheme, the BF-IBE scheme, based on bilinear pairings on elliptic curves in 2001. Goyal [26] introduced the Accountable Authority Identity-Based Encryption (A-IBE) scheme to reduce trust in the PKG as an effective solution for the key escrow problem. The A-IBE scheme was matured into the Black Box AIBE scheme by Goyal et al. [27]. Later, Garg et al. [28, 29] proposed the
Registration-Based Encryption (RBE) scheme, aggregating the KGC and compactly compressing all users’ public keys into the master public key.

1.3. Organization. To begin, in Section 2, we presented helpful preliminary information. Then, in Section 3, we detailed the construction of our approach. In Section 4, we analyzed the security features of the scheme. In Section 5, the IND-CPA security of our scheme was proved. In Section 6, we analyzed the scheme’s performance and compared it with other schemes in performance and security. Finally, we concluded with comments on our work and limitations for future work in Section 7.

2. Preliminaries

We briefly introduced the background knowledge needed to read this paper, including Symmetric Bilinear Function, BDH Problem, X3DH Algorithm, BF-IBE Scheme, Hess’s IBS Scheme, and RIBE Scheme.

2.1. Notation. We defined a secure symmetric encryption algorithm as Enc. For Enc $(M_1, M_2, M_3, K)$, $K$ is the key, and $M_1, M_2, M_3$ are the messages to be encrypted. The symmetric encryption algorithm Dec and the signature algorithm Sign are given similar forms.

2.2. Symmetric Bilinear Function. Let $q$ be a large prime, $G_1$ is an additive group, and $G_2$ is a multiplicative group of the same order $q$. Given a symmetric bilinear function from $G_1$ to $G_2$ as $e: G_1 \times G_1 \rightarrow G_2$ and $e$ satisfies the following properties:

- Bilinearity: $\forall P, Q \in G_1$ and $a, b \in \mathbb{Z}$, such that $e([a]P, [b]Q) = e(P, Q)^{ab}$.
- Nondegeneracy: $\forall P \in G_1$, such that $e(P, P) \neq 1$, that means the generator of $G_2$ cannot be the identity element.
- Computability: $\forall P, Q \in G_1$, there exists an effective algorithm to compute $e(P, Q)$.
- Symmetry: $\forall P, Q \in G_1$, such that $e(P, Q) = e(Q, P)$.

2.3. BDH Problem. Given $(P, [a]P, [b]P, [c]P) \in G_2$, compute $e(P, P)^{abc} \in G_2$, where $e$ is a bilinear function, $P$ is the generator on $G_1$, and $G_1, G_2$ are two groups of the same order $q$[30]. Assume the algorithm $A$ is used to solve the BDH problem:

$$\Pr[A(P, [a]P, [b]P, [c]P) = e(P, P)^{abc}] \geq \varepsilon.$$  

2.4. X3DH Algorithm. X3DH [31] is a tripartite key negotiation protocol where the two communicating parties go through a server to achieve asynchronous encrypted communication.

Figure 2 shows the specific design of the protocol. First, Bob registers his public key bundle to the server. When Alice wants to establish communication with Bob, she retrieves Bob’s public key bundle from the server. Then, Alice verifies the prekey signature and performs four Diffie–Hellman key exchanges in the order shown in Figure 3. Finally, she generates the shared key $SK$. Bob similarly generates $SK$, decrypts the message $M_1$ and Alice’s safety number $SN$, and encrypts the message $M_2$ and the locally calculated safety number $SN$. The server forwards it to Alice.

Figure 4 shows an example when the protocol is subjected to a MITM attack by a malicious server Trudy. Both parties in the protocol can get the safety number forwarded to each other by the server. However, it is still impossible to determine whether there is a MITM attack based on direct comparison results. Through the original channel comparison, the safety number will still be tampered with by the man in the middle.

2.5. BF-IBE Scheme. The public key of the BF-IBE [16] scheme is determined by the identity, and a trusted third-party PKG generates the private key.

- Setup: Suppose $k$ is the security parameter, $q$ is a large prime of $k$-bits, $G_1$ is an additive group, and $G_2$ is a multiplicative group where the order of both groups is $q$, $e: G_1 \times G_1 \rightarrow G_2$ is a bilinear mapping. $P$ is a generator of $G_1$, $H_1: [0, 1]^* \rightarrow G_1^*$ and $H_2: [0, 1]^n \rightarrow [0, 1]^n$ are two hash functions where $n$ is the length of the message to be encrypted. The PKG picks $s \in \mathbb{Z}_q^*$ as the master key and computes the public key $P_{pub} = [s]P$. PKG keeps the public system parameter $\text{param} = \{q, G_1, G_2, e, n, P, P_{pub}, H_1, H_2\}$ and the secret $s$.

- Key generation: The public key corresponding to $ID \in [0, 1]^*$ is $Q_{ID} = H_1(ID)$. The PKG computes the private key $S_{ID} = [s]Q_{ID}$.

- Encryption: The public key of the receiver is $Q_{ID}$. To encrypt $M \in [0, 1]^*$, calculate $r^eZ_q^* \subseteq g = e(Q_{ID}, P_{pub}) \in G_2^*$, $C = ([r]P, M \oplus H_2(g^e))$.

- Decryption: For the given ciphertext $C = [U, V]$ and the private key $S_{ID}$, the receiver can decrypt $C$ as $M = V \oplus H_2(e(S_{ID}, U))$.

2.6. Hess’s IBS Scheme. The signature scheme used in this paper is Hess’s IBS scheme [32], and Hess gives the security proof of their scheme under the random oracle model. The scheme reduces the number of bilinear pair operations and outperforms Shamir’s scheme [33] and Jae Cha’s scheme [34] in terms of efficiency.

- Setup: Suppose $q$ is a large prime, $G_1$ is an additive group, and $G_2$ is a multiplicative group where the order of both groups is $q$, $e: G_1 \times G_1 \rightarrow G_2$ is a bilinear mapping. $P$ is a generator of $G_1$, $H_1: [0, 1]^* \rightarrow G_1^*$ and $H_2: [0, 1]^* \rightarrow Z_q^*$ are two hash functions. The PKG picks $s \in \mathbb{Z}_q^*$ as the master key and computes the public key $P_{pub} = [s]P$. The PKG keeps the system parameter $\text{param} = \{q, G_1, G_2, e, P, P_{pub}, H_1, H_2\}$ public and $s$ secret.

- Key generation: After the identity verification passes, the PKG computes the public key $Q = H_1(ID)$ based
2.7. RIBE Scheme. To make the IBE algorithm support efficient public key revocation, Boldyreva [18] et al. proposed the RIBE scheme. Each user corresponds to two attributes: identity and validity time of the public key, and therefore corresponds to two keys in RIBE: the private key and the key update. The core of RIBE is that only having two private keys can decrypt the message, while the PKG cannot update the key update to the user whose public key has been revoked. We only present the basic definition of RIBE here. For more details, please consult [18].

Setup: $S(1^n, n) \rightarrow (pk, mk, rl, st)$: $n$ is the number of users, $pk$ is the public parameters, $mk$ is the master key, $rl$ is the revocation list, and $st$ is the binary tree representing states.

Private key generation: $SK(pk, mk, \omega, st) \rightarrow (sk_\omega, st)$: $\omega$ is the identity, $sk_\omega$ is the private key, and the state $st$ is updated by this function.

Key update generation: $KU(pk, mk, tr, rl, st) \rightarrow ku_t$: $t$ is the validity time and $ku_t$ is the key update.

Decryption key generation: $DK(sk_\omega, ku_t) \rightarrow dk_{\omega,t}$ or $\perp$: If the identity $\omega$ is revoked, then the function outputs $\perp$; else outputs the decryption key $dk_{\omega,t}$.

Encryption: $E(pk, \omega, t, m) \rightarrow c$: $t$ is when the encryption occurs, $m$ is the plaintext, and $c$ is the ciphertext.

Decryption: $D(dk_{\omega,t}, c) \rightarrow m$ or $\perp$: If the ciphertext $c$ is invalid, the function outputs $\perp$; else outputs the plaintext $m$.

Revocation: $R(\omega, t, rl, st) \rightarrow rl$: $\omega$ is the identity to be revoked, $t$ is the revocation time, and $rl$ is updated by this function.

3. Scheme Definition

We first define the threat model of our IBE-Signal scheme, then introduce the framework of the whole scheme, and finally present the scheme’s construction in two parts.
KGC sets most system parameters, and the CPCs also have the private key share for risk reduction. The Key Authority is fully trusted and only responsible for public key revocation. In addition, our public key revocation feature is optional, and if the user turns it off, the Key Authority will no longer serve him.

We assume that an ambitious adversary can collude with the KGC and \( n-1 \) CPCs. In other words, at least one CPC in our scheme is credible. Corrupt KGC/\( n-1 \) CPCs can assign the partial private key to any identity submitted by the adversary. The adversary can launch a chosen-plaintext attack and can finally select an \( ID^* \) to challenge. We allow the adversary to retrieve the partial private key from the KGC and \( n \) partial private keys from the CPCs in this case. This process can be repeated polynomial bounded times.

Even with these capabilities, the adversary has a negligible advantage against our IBE-Signal scheme.

3.2. The Framework of the IBE-Signal Scheme. Like the SSL protocol [35], our IBE-Signal scheme contains parts of the SESSION and the CONNECTION. The SESSION part refers to a collection of parameters and encryption keys generated through a handshake between two communicating parties. In contrast, the CONNECTION part refers to a subsequent session established after the SESSION part using the shared key \( RK \). The SESSION part is more overhead than the CONNECTION part for the server. Once the SESSION part is established, a user can use a SESSION part context to create the CONNECTION part for the next time. This reduces the communication overhead of the protocol. The SESSION part consists of eight Probabilistic Polynomial-Time algorithms and can be described as \( \text{SESSION} = (S,EKG,SI,E,DKG,D,V,R) \). The CONNECTION part consists of four
Probabilistic Polynomial-Time algorithms and can be described merely as \textit{CONNECTION} \(\Rightarrow (\text{SKG}, E, \text{RKG}, D)\).

The \textit{SESSION} part is shown in Figure 7, and the \textit{CONNECTION} part is shown in Figure 8.

3.3. Scheme Construction. We present the scheme's construction in two parts, the \textit{SESSION} and the \textit{CONNECTION}. Since the scheme uses many symbols, Table 2 provides a table of characters for the reader's convenience.

\begin{equation}
(q, G_1, G_2, e)\rightarrow G, P \rightarrow G_1, S_0 \rightarrow Z_q^*, Y_0 = P_0 = [S_0]P,
H_1: [0, 1]^* \rightarrow G_1^*, H_2: G_2 \rightarrow [0, 1]^k, H_3: [0, 1]^k \rightarrow [0, 1]^k,
H_4: [0, 1]^* \rightarrow Z_q^*, H_5: [0, 1]^* \times G_2 \rightarrow Z_q^*,
H_6: [0, 1]^k \times [0, 1]^k \rightarrow [0, 1]^k, \text{KDF: } G_1 \times [0, 1]^k \rightarrow [0, 1]^k,
\end{equation}

where \(P\) is a generator of group \(G_1\), \(S_0\) is the private key of the KGC, \(P_0\) is the public key of the KGC, and \(Y_0\) is the shared public key of the KGC. \(H_1, H_2, H_3, H_4, H_5, H_6, \text{KDF}\) are secure hash functions, \(k\) represents the length of the message key. The system parameter of the KGC \(\text{param}_1 = (q, G_1, G_2, e, k, P, P_0, Y_0, H_1, H_2, H_3, H_4, H_5, H_6, \text{KDF})\) is public, while the private key of the KGC \(S_0\) is private.

(2) \textit{CPC Setup} (\text{param}_1) \rightarrow (\text{param}_2, S_i), \text{where } i = 1, 2, \ldots, n. \text{This algorithm is run by the CPCs:}

Each CPC\(_i\) chooses \(S_i \in Z_q^*\) as the private key and computes \(P_i = [S_i]P\) as the public key, where \(i = 1, 2, \ldots, n, n\) is the number of the CPCs. Each CPC\(_i\) computes \(Y_i = [S_i]Y_{i-1}\) as the shared public key. Specially, we define \(Y_n = Y\). For example, the CPC\(_1\) can compute \(Y_1\) with the private key \(S_1\) and the shared public key \(Y\) of the KGC. We use this chain from the KGC to the CPC\(_n\) to compute the shared public key \(Y_i\) of each entity. It is important to emphasize that when the CPC\(_{i-1}\) sends \([Y_0, \ldots, Y_{i-1}]\) to the CPC\(_n\) then the CPC\(_i\) must verify the correctness of \([Y_0, \ldots, Y_{i-1}]\). Take the CPC\(_3\) as an example. The CPC\(_3\) verifies \(Y_2\) by computing \(e(Y_0, P_1) = e(Y_1, P)\) and \(e(Y_1, P_3) = e(Y_2, P)\). System parameter of the CPC\(_3\) \text{param}_2 = (P_3, Y_i) is public while the private key of the CPC\(_i\) \(S_i\) is private, where \(i = 1, 2, \ldots, n, n\).

(3) \textit{Key Authority Setup} (\text{param}_1, nu) \rightarrow (\text{param}_3, a, rl, st). \text{This algorithm is run by the Key Authority:}

\begin{equation}
g = P, a \rightarrow Z_q^*, g_1 = [a]g, g_2, h_1, h_2, h_3 \rightarrow G_1^*,
\end{equation}

where \(nu\) is the number of users, \(rl\) is the initially empty revocation list, and \(st\) is the perfect binary tree with \(nu (nu\) is even) or \(nu + 1 (nu\) is odd) leaf nodes representing states. Let \(\tau\) be the minimum time interval. System parameter of the Key Authority \text{param}_3 = (q, g_1, g_2, h_1, h_2, h_3, a, rl, st)\) is public, while the private key of the Key Authority \(a\) is private. In addition, the following two operations are defined just as [18]. For \(x, j \in Z\), set \(j \equiv Z\), the Lagrange coefficient \(\Delta_{i,j}(x)\) is defined as follows:

\begin{equation}
\Delta_{i,j}(x) = \prod_{j \neq i} \left( \frac{x - j}{i - j} \right).
\end{equation}

For \(x \in Z, j \equiv Z, g, h_1, \ldots, h_t \in G_1,\)

\begin{equation}
H_{g,i,h_1,\ldots,h_t}(x) = \left( [x^2]g \right) \prod_{i=1}^{t} \left( [\Delta_{i,j}(x)] h_i \right).
\end{equation}

In the \textit{Setup} step, the final public parameter is \text{params} = (\text{param}_1, \text{param}_2, \text{param}_3).

\text{Encryption Key Generation} \text{EKG}(\text{params}, \text{ID}_B, \text{time}) \rightarrow (r, \text{RK}, e, U, \text{rc}), \text{including the following two algorithms:}

(1) \textit{Encryption RootInput Generation} (\text{params}, \text{ID}_B) \rightarrow (r, U, \text{RootInput}). \text{This algorithm is run by the sender:}

Parse params as \((H_1, H_2, P, Y, q, G_1, e)\).

\begin{equation}
Q_{ID_B} = H_1(\text{ID}_B) \in G_1^*,
\end{equation}

\begin{equation}
r \rightarrow Z_q^*, U = [r]P, \text{RootInput} = H_2(\left( e(Q_{ID_B}, Y) \right)^x),
\end{equation}

where \(ID_B\) is the receiver's identity, \text{RootInput} is the input of the root chain in the Double Ratchet algorithm.

(2) \textit{Encryption Revocation Key Generation} (\text{params}, \text{ID}_B, \text{time}) \rightarrow (r_k, \text{rc}). \text{This algorithm is run by the sender:}
1. **Setup**
   - 1.1 KGC Setup
   - 1.2 CPC Setup
   - 1.3 Key Authority Setup

2. **Encryption Key Generation → (RK, ek, U, rc)**
   - 2.1 Encryption RootInput Generation
   - 2.2 Encryption Revocation Key Generation

3. **Sign → σ**
   - 3.1 Sender's Private Key Generation → $S_{ID_A}$
   - 3.2 Signature Generation → σ

4. **Encrypt (ek, U, rc, T, M, σ) → C**
   - V = Enc (M || T || σ, ek)
   - C = (U, V, rc)

5. **Decryption Key Generation → (RK, ek)**
   - 5.1 Receiver's Private Key Generation → $S_{ID_B}$
   - 5.2 Decryption RootInput Generation → $S_{ID_B}$
   - 5.3 Revocation Private Keys Generation
   - 5.4 Revocation Key Updates Generation
   - 5.5 Revocation Key Selection
   - 5.6 Decryption Revocation Key Generation

6. **Decrypt (C, ek) → (T, M, σ)**

7. **Verify (params, σ, M, T, ID_A, T') → true or false**

8. **Revocation (w, t, rl, st) → rl**

**Figure 7:** The SESSION part of the IBE-Signal scheme.

**Figure 8:** The CONNECTION part of the IBE-Signal scheme.
Table 2: Abbreviations and notations.

<table>
<thead>
<tr>
<th>Abbreviations/notations</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>Secure parameter</td>
</tr>
<tr>
<td>$q$</td>
<td>Big prime</td>
</tr>
<tr>
<td>$G_1$</td>
<td>Additive group on an elliptic curve of the order $q$</td>
</tr>
<tr>
<td>$G_2$</td>
<td>Multiplicative group on an elliptic curve of the order $q$</td>
</tr>
<tr>
<td>$e: G_1 \times G_1 \rightarrow G_2$</td>
<td>Bilinear function</td>
</tr>
<tr>
<td>$P, g, g_2, h_1, h_2, h_3$</td>
<td>Generator of the group $G_1$</td>
</tr>
<tr>
<td>$S_0, P_0, Y_0$</td>
<td>The private key, the public key, and the shared public key of the KGC</td>
</tr>
<tr>
<td>$S_i, P_i, Y_i$ ($i = 1, 2, ..., n$)</td>
<td>The private key, the public key, and the shared public key of the $i^{th}$ CPC</td>
</tr>
<tr>
<td>$Y$</td>
<td>The shared public key of the CPC$_n$ that equals to $Y_n$</td>
</tr>
<tr>
<td>$H_1: [0, 1]^* \rightarrow G_1^*$</td>
<td>A secure hash function that takes a string of any length as input and gives an element on $G_1$</td>
</tr>
<tr>
<td>$H_2: G_2 \rightarrow [0, 1]^k$</td>
<td>A secure hash function that takes an element on $G_2$ and gives a string of length $k$</td>
</tr>
<tr>
<td>$H_3: [0, 1]^* \times G_2^* \rightarrow Z_q^*$</td>
<td>A secure hash function that takes a string of any length and an element on $G_2$ and gives an element on $Z_q^*$</td>
</tr>
<tr>
<td>$H_4: [0, 1]^k \times [0, 1]^k \rightarrow [0, 1]^k$</td>
<td>A secure hash function that takes two strings of length $k$ and gives a string of length $k$</td>
</tr>
<tr>
<td>$KDF: G_1 \times [0, 1]^k \rightarrow [0, 1]^k$</td>
<td>Secure key derivation function that takes an element on $G_1$ and a string of length $k$ and gives a string of length $k$</td>
</tr>
<tr>
<td>$nu$</td>
<td>Number of users</td>
</tr>
<tr>
<td>$rl$</td>
<td>Initially empty revocation list</td>
</tr>
<tr>
<td>$st$</td>
<td>A binary tree with at least $nu$ leaf nodes representing states</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Minimum time interval</td>
</tr>
<tr>
<td>$a$</td>
<td>The private key of key authority</td>
</tr>
<tr>
<td>$g_1$</td>
<td>Part of the system parameter of key authority and $g_1 = [a]g$</td>
</tr>
<tr>
<td>$\text{param}_1$</td>
<td>System parameter of the KGC</td>
</tr>
<tr>
<td>$\text{param}_2$</td>
<td>System parameter of the CPCs</td>
</tr>
<tr>
<td>$\text{param}_3$</td>
<td>System parameter of key authority</td>
</tr>
<tr>
<td>$\text{params}$</td>
<td>The final public parameter in the setup step</td>
</tr>
<tr>
<td>$ID_R \in {0, 1}^*$</td>
<td>Identity of the receiver</td>
</tr>
<tr>
<td>$ID_S \in {0, 1}^*$</td>
<td>Identity of the sender</td>
</tr>
<tr>
<td>$Q_{ID_R} \in G_1^*$</td>
<td>Mapping of $ID_R$ on $G_1$</td>
</tr>
<tr>
<td>$Q_{ID_S} \in G_1^*$</td>
<td>Mapping of $ID_S$ on $G_1$</td>
</tr>
<tr>
<td>$S_{ID_R} \in G_1^*$</td>
<td>The private key of the receiver</td>
</tr>
<tr>
<td>$S_{ID_S} \in G_1^*$</td>
<td>The private key of the sender</td>
</tr>
<tr>
<td>$\omega$</td>
<td>The time when the encryption occurs</td>
</tr>
<tr>
<td>$t$</td>
<td>Mapping of time on $Z_q^*$</td>
</tr>
<tr>
<td>$\text{RootInput}$</td>
<td>The input of the root chain in the double ratchet algorithm</td>
</tr>
<tr>
<td>$rk$</td>
<td>Revocation key</td>
</tr>
<tr>
<td>$RK$</td>
<td>The input of the root chain in the CONNNECTION part of the IBE-signal scheme</td>
</tr>
<tr>
<td>$ek$</td>
<td>Message key for the encryption of the first message</td>
</tr>
<tr>
<td>$x$</td>
<td>Blinding factor in the private key issuance process</td>
</tr>
<tr>
<td>$T$</td>
<td>Timestamp used to prevent replay attacks</td>
</tr>
<tr>
<td>$T'$</td>
<td>Receiver’s local time</td>
</tr>
<tr>
<td>$M$</td>
<td>The message that needs to be signed</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Signature</td>
</tr>
<tr>
<td>$sk_w$</td>
<td>Private keys for revocation</td>
</tr>
<tr>
<td>$ku_w$</td>
<td>Key updates for revocation</td>
</tr>
<tr>
<td>$dk_w$</td>
<td>Revocation key selected from $sk_w$ and $ku_w$</td>
</tr>
<tr>
<td>$PK_B, SK_B$</td>
<td>Bob’s public and private keys in Diffie–Hellman key exchange in CONNNECTION part</td>
</tr>
<tr>
<td>$PK_A, SK_A$</td>
<td>Alice’s public and private keys in Diffie–Hellman key exchange in CONNNECTION part</td>
</tr>
<tr>
<td>$S_{\text{Session}, K_j}$</td>
<td>The $j^{th}$ message key of the sender in the $i^{th}$ session</td>
</tr>
<tr>
<td>$R_{\text{Session}, K_j}$</td>
<td>The $j^{th}$ message key of the receiver in the $i^{th}$ session</td>
</tr>
</tbody>
</table>
Parse params as \((H_3, H_4, H_6, q, k, g, g_1, g_2, h_1, h_2, h_3, e)\). 
\[
\begin{align*}
\omega &= H_4(1D_B) \in \mathbb{Z}_q^*, t = H_4(\text{time}) \in \mathbb{Z}_q^*, \mathbb{Z}_q^*, \mathbb{Z}_q^*, \\
c_1 &= rk \cdot (e(g_1, g_2))^t, c_2 = [z]g, \\
c_\omega &= [z]H_{g_1, h_1, h_2, h_3}(\omega), c_t = [z]H_{g, h_1, h_2, h_3}(t),
\end{align*}
\]
(7)

where time is the time when the encryption occurs and \(rk\) is the revocation key. Let \(J\) be \([1, 2, 3]\) and \(rc\) be \((\omega, t, c_\omega, c_1, c_2)\).

In the Encryption Key Generation step, the final encryption key \(ek\):
\[
RK = H_6(\text{RootInput}, rk), ek = H_3(RK).
\]
(8)

Sign \(SI\) (params, \(S_i, ID_A, T, M\) \longrightarrow \sigma, including the following two algorithms:

(1) Sender’s Private Key Generation (params, \(S_i, ID_A\) \longrightarrow \(S_{ID_A}^i\), where \(i = 0, 1, \ldots, n\). The sender runs this algorithm and interacts with the KGC and the CPCs, as shown in Figure 10.

Parse params as \((q, e, P, Y, H_2)\).

The sender does the following steps: \(x \leftarrow \mathbb{Z}_q^*, x = [x]P, Q_{ID_A} = H_1(ID_A), D_{ID_A} = [x]Q_{ID_A}\), where \(ID_A\) is the sender’s identity and \(x\) is the blinding factor. We use the blind signature to protect the sender’s partial private key from being obtained by an attacker or server. The sender sends \((ID_A, D_{ID_A}, X)\) to the KGC. It should be noted that the sender must go through the corresponding identity authentication process, such as password authentication, to get the signed private key, whether interacting with the KGC or the CPCs.

\begin{align*}
x &\leftarrow \mathbb{Z}_q^*, \\
x &= [x]P, \\
Q_{ID_A} &= H_1(ID_A), \\
D_{ID_A} &= [x]Q_{ID_A},
\end{align*}

Since this is not our focus, this paper has no specific design.

The KGC and the CPCs do the following steps: the KGC computes \(Q_{ID_A} = H_1(ID_A)\) and verifies the correctness of \(e(Q_{ID_A}, X) = e(D_{ID_A}, P)\). This ensures that the KGC is issuing the partial private key to \(ID_A\). The KGC computes \(D_0 = [S_0]D_{ID_A}, X_0 = [S_0]X\) and sends them to the sender. The sender sends \((ID_A, D_{ID_A}, X_{i-1})\) to the CPCi. The CPCi computes \(Q_{ID_A} = H_1(ID_A)\) and verifies the correctness of \(e(Q_{ID_A}, X_{i-1}) = e(D_{ID_A}, P)\), then calculates \(D_i = [S_i]D_{ID_A}, X_i = [S_i]X_{i-1}\) and sends them to the sender where \(i = 1, 2, \ldots, n\). After the iterations, the sender can get \(D_n = [S_n]D_{n-1}\).
The sender extracts the private key by computing $S_{ID_a} = [x^{-1}]D_{ID_a}$, then verifies the correctness of $e(S_{ID_a}, P) = e(Q_{ID_a}, Y)$. (2) Signature Generation \((params, T, M, S_{ID_a}) \rightarrow \sigma\). This algorithm is run by the sender: 

\[
\begin{align*}
\text{Parse params as } (q, e, G_1, P, H_2). \\
b = \frac{Z_q^*}{O^{e} G_1}, R = (e(O, P))^b, \quad y = H_3(M || T, R), F = [y]S_{ID_a} + [b]O, \sigma = (F, y),
\end{align*}
\]

where \(T\) is the timestamp used to prevent replay attacks, \(M\) is the message that needs to be signed, and \(\sigma\) is the signature. In the SESSION part, \(M\) can be a fixed format message shown in Figure 11, such as "Hi! I’m Alice." When Alice friends Bob, a notification is automatically sent when the initial session is established.

**Encrypt** \(E(ek, U, rc, T, M, \sigma) \rightarrow C\). This algorithm is run by the sender: 

Let **Enc** be a secure symmetric encryption algorithm. 

\[
V = \text{Enc}(M || T || \sigma, ek), C = (U, V, rc),
\]

where \(C\) is the ciphertext sent to the receiver.

### Decryption Key Generation

\(DKG\) (params, \(a_S, S_{ID_B}\), time, \(C, st, rl\)) \(\rightarrow (RK, ek)\), including the following six algorithms:

1. **Receiver’s Private Key Generation** \((params, S_{ID_B}) \rightarrow S_{ID_a}\). The receiver runs this algorithm and interacts with the KGC and the CPCs: The function is like Sender’s Private Key Generation, so we will not go over it here.

2. **Decryption RootInput Generation** \((params, S_{ID_a}, C) \rightarrow \text{RootInput}\). This algorithm is run by the receiver: 

\[
\text{Parse params as } (H_2, e) \text{ and } C = (U, V, rc),
\]

\[
\text{RootInput} = H_2(e(S_{ID_a}, U)).
\]

3. **Revocation Private Keys Generation** \((params, a, S_{ID_B}, st) \rightarrow (sk_w, st)\). This algorithm is run by the Key Authority: 

\[
\omega = H_4(ID_B) \in Z_q^*.
\]

Select an empty leaf node \(v\) from \(st\) and store \(\omega\) in it. In addition, the **Path** function is defined as follows: the set of all nodes on the path from the leaf node to the root node, including that leaf node and the root node.

\[
\begin{align*}
&\forall i \in \text{Path}(v) \\
&(i) \text{ if } a_i \text{ is undefined,} \\
&(ii) \text{ then } a_i = \frac{Z_q^*}{g_i} \text{ and store } a_i \text{ in node } i, \\
&(iv) \text{ compute } r_j = [Z_q^*], d_i = [r_j]g_i, \\
&(v) \text{ } D_i = ([a_i t + a_i]g_i) + ([r_j]H_{g_i, J, h_i, h_i, h_i}(t)).
\end{align*}
\]

After the above code is executed, we get the private keys for revocation \(sk_w\) as \(\{(i, d_i, D_i)\}_{(i \in \text{Path}(v))}\) and the updated state tree \(st\). (iii) $D_i = ([a_i t + a_i]g_i) + ([r_j]H_{g_i, J, h_i, h_i, h_i}(t))$. (v) $D_i = ([a_i t + a_i]g_i) + ([r_j]H_{g_i, J, h_i, h_i, h_i}(t))$. (vi) $a_i = \frac{Z_q^*}{g_i}$ and store $a_i$ in node $i$,

After the above code is executed, we get the **private keys for revocation** $sk_w = \{(i, d_i, D_i)\}_{(i \in \text{Path}(v))}$ and the updated state tree $st$. (v) $D_i = ([a_i t + a_i]g_i) + ([r_j]H_{g_i, J, h_i, h_i, h_i}(t))$. (vi) $a_i = \frac{Z_q^*}{g_i}$ and store $a_i$ in node $i$,
The revoked node
The node output by the KUNodes function

Figure 12: The KUNodes tree when node #7 is revoked.

(iv) return $dk_{a,i}^{r}$.

(6) Decryption Revocation Key Generation
$(dk_{a,i}, C) \rightarrow rk$. This algorithm is run by the receiver:
Parse $dk_{a,i}$ as $(d_i, D_i, e_i, E_i), C$ as $(U, V, rc)$ and $rc$ as $(\omega, t, c_{\omega}, c_1, c_2)$.

$$rk = c_1 \left( e(d_i, c_\omega) \right)^{t/w} \left( e(e_i, c_1) \right)^{w/u-t} \cdot (14)$$

In the Decryption Key Generation step, the final decryption key:

$$RK = H_e(\text{RootInput}, rk), ek = H_3(RK). \quad (15)$$

Decrypt $D(C, ek) \rightarrow (T, M, \sigma)$. This algorithm is run by the receiver:
Parse $C$ as $(U, V, rc), V$ as $\text{Enc}(M|T|\sigma, ek)$. Let Dec be a secure symmetric decryption algorithm.

$$M|T|\sigma = \text{Dec}(V, ek). \quad (16)$$

Verify $V$ (params, $\sigma, M, T, ID_A, T'$) $\rightarrow$ true or false. This algorithm is run by the receiver:
Parse params as $(e, P, Y, H_1, H_5, \tau), \sigma$ as $(F, y)$. Let $T'$ be the receiver's local time.

$$Q_{ID_A} = H_4(ID_A), \quad R = e(F, P)(e(Q_{ID_A}, Y))^{-w}. \quad (17)$$

(i) if $H_5(M|T, R) = y \& \& T' - T \leq \tau$ then return true
(ii) else return false.

Revocation $R(a, t, rl, st) \rightarrow rl$. This algorithm is run by the Key Authority:
(i) $\forall v \in \left\{ \text{leaf f nodes in st with } a \right\}$
(ii) $\text{add}(v, t)$ to rl
(iii) return rl.

3.3.2. CONNECTION Part. In the CONNECTION part, we show how the first two messages are sent and received in the first session.
4. Security Features’ Analysis

This paper selects two features to analyze in detail: resisting MITM attacks and supporting public key revocation.

4.1. Our IBE-Signal Scheme Can Resist MITM Attack. We discussed that MITM attacks cannot be implemented in both the SESSION part and the CONNECTION part.

In the SESSION part, mutual authentication exists. To indicate the existence of mutual authentication in the SESSION part, we show a simplified version of our IBE-Signal protocol in Figure 13. Based on the characteristics of the IBE scheme, Alice can get Bob’s public key \( Q_{DB_j} \) directly from Bob’s ID and then encrypt the message \( M \) by the public key \( Q_{DB_j} \). \( [r]P \) is a random challenge to Bob’s identity. The only way to get \( S_{Session1, K1} \) is to have Bob’s private key, which encrypts message \( M_1 \) as a response to the challenge \( [r]P \). This completes the authentication of Bob and is resistant to replay attacks. Authentication of Alice is done by signature and timestamp and prevents replay attacks. A MITM attack is not possible when mutual authentication is established.

In the CONNECTION part, launching a MITM attack is ineffective.

Since we use the Diffie–Hellman ratchet in the CONNECTION part, an attacker can launch a MITM attack to hijack the session, as shown in Figure 14. However, both parties have negotiated RK in the SESSION part, while the attacker has no RK. The attacker cannot calculate \( K1 \) and \( K2 \) and naturally cannot send the message \((PK_{T1}, Enc(M_1, K2))\).

4.2. Our IBE-Signal Scheme Supports Public Key Revocation. Once the user’s public key expires, the Key Authority will add it to \( rl \), and the KUNodes function will output a new point set. As shown in Figure 12, the set consisting of red dots is \( \{ (i, d_i, D_i) \}_{i \in \text{Path}(v)} \) and the set consisting of green dots is \( \{ (j, e_j, E_j) \}_{k \in \text{KUNodes}(\text{st}, T, t_{l, t})} \). The two sets have no intersection and thus cannot generate the revocation key \( dk_{u,t} \).

Assume that the encryption time is \( t \) and the expiration time is \( t' \). Formula 22 shows that the attacker cannot get \( rk \) using the old revocation key \( dk_{u,t} \), so the initial session cannot be established.

\[
\begin{align*}
C & = (\{r\}P, Enc(T || M || \sigma, ek, rc)) \\
C_1 & = (PK_{T1}, Enc(M_1, S_{Session1, K1}))
\end{align*}
\]
5. Security Proof

We first prove the correctness of the scheme, then introduce the security model, and subsequently show that the scheme is IND-ID-CPA secure.

**Theorem 1.** Our IBE-Signal scheme satisfied the correctness property.

**Proof.** To facilitate the proof, we distinguish the corresponding recipient symbols by adding ",", e.g. `RootInput` and `RootInput'`.

**Lemma 1.** The SESSION part satisfied the correctness property.

**Proof**
Lemma 2. The CONNECTION part satisfied the correctness property.

\[
M_1 = \text{Dec}(\text{Enc}(M_1, S_{\text{Session1-K1}}), R_{\text{Session1-K1}})
\]
\[
= \text{Dec}(\text{Enc}(M_1, S_{\text{Session1-K1}}), H_3(\text{KDF}(\text{DHkey}_R_{\text{Session1-K1}}, RK)))
\]
\[
= \text{Dec}(\text{Enc}(M_1, S_{\text{Session1-K1}}), H_3(\text{KDF}([r]PK_{B1}, RK)))
\]
\[
= \text{Dec}(\text{Enc}(M_1, S_{\text{Session1-K1}}), H_3(\text{KDF}([rSK_{B1}]P, RK)))
\]
\[
= \text{Dec}(\text{Enc}(M_1, S_{\text{Session1-K1}}), H_3(\text{KDF}([SK_{B1}]|rP), RK)))
\]
\[
= \text{Dec}(\text{Enc}(M_1, S_{\text{Session1-K1}}), H_3(\text{KDF}([SK_{B1}]U, RK)))
\]
\[
= \text{Dec}(\text{Enc}(M_1, S_{\text{Session1-K1}}), H_3(\text{KDF}(\text{DHkey}, S_{\text{Session1-K1}})))
\]
\[
= \text{Dec}(\text{Enc}(M_1, S_{\text{Session1-K1}}), S_{\text{Session1-K1}})
\]
\[
= M_1. 
\]

Proof
Lemma 3. The SignVerify part satisfied the correctness property.

Proof

\[
y' = H_5(M||T, R') = H_5(M||T, e(F, P)e(Q_{ID_1}^* - Y)^\beta)
\]

\[
= H_5(M||T, e([y]_{SI_{ID_1}} + [b]O, P)e(Q_{ID_1}^* - Y)^\beta)
\]

\[
= H_5(M||T, e([y]_{SI_{ID_1}} - P)e([b]O, P)e(Q_{ID_1}^* - Y)^\beta)
\]

\[
= H_5(M||T, e(Q_{ID_1}^*, Y)^\beta e(O, P)^b e(Q_{ID_1}^* - Y)^\beta)
\]

\[
= H_5(M||T, e(O, P)^b)
\]

\[
= y.
\]

(25)

5.1. Security Model. It should be noted that the proof of the IND-sRID-CPA security of the RIBE scheme is given in [18], so the revocation key rk is provided directly by the challenger in the Setup phase of our game. The next game describes the security model of our scheme:

Setup. The challenger inputs a secure parameter \( \kappa \) and outputs public parameters \( \text{params} \), revocation key \( \text{rk} \), and private keys \( a_i \) and \( S_i \) where \( i = 0, 1, 2, \ldots, n \). The challenger sends \( \text{params} \) and \( \text{rk} \) to the adversary. The challenger controls the following queries and can be queried by the adversary.

- \( H_1 \) queries: Given identity \( ID \), the challenger runs these queries and gives \( Q_{ID} \).
- \( H_2 \) queries: Given \( F \in G_2 \), the challenger runs these queries and gives \( H_j \in \{0, 1\}^* \).

Phase 1. The adversary chooses an ID and sends queries of partial private keys of the ID. The challenger generates the partial private keys \( \{S_{ID} = \prod_{i=0}^{\lambda} S_i Q_{ID}\}_{i=0,1,\ldots,n} \) and then sends them to the adversary.

Challenge. The adversary outputs two plaintexts \( M_0, M_1 \) of equal length and the identity \( ID^* \) intentionally to challenge. The only restriction is that \( ID^* \) does not appear in any query in Phase 1. The adversary sends \( M_0, M_1 \) and \( ID^* \) to the challenger. The challenger randomly chooses \( k \) as the honest CPC’s serial number and generates the partial private key \( S_{ID'} = \prod_{i=0, k \neq k} S_i Q_{ID'} \). The challenger randomly chooses a bit \( \beta^k \{0, 1\} \) and computes \( C^* = \text{Enc}(M^*, S_{ID'}) \) where \( S_{ID'} = \prod_{i=0, k \neq k} S_i Q_{ID'} \) then sends \( \{S_{ID'}\}_{i=0,1,\ldots,n,i \neq k} \) and \( C^* \) to the adversary.

Phase 2. The adversary sends queries of partial private keys of \( ID (ID \neq ID^*) \) to the challenger. The challenger responds in the way of Phase 1.

Guess. The adversary outputs his guess \( \beta' \in \{0, 1\} \). If \( \beta' = \beta \), the adversary attacks the IBE-Signal scheme successfully.

The adversary’s advantage is defined as the function of the safe parameter \( \kappa \):

\[
\text{Adv}_{A, \text{IBE-Signal}}^{\text{IND-ID-CPA}}(\kappa) = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|.
\]

(26)

Theorem 2. Assume \( H1 \) and \( H2 \) are random oracle models, and the BDH problem is infeasible to solve; our IBE-Signal scheme is semantically secure against IND-ID-CPA attack.

Proof. Expressly, assume the adversary \( A \) can run at most \( q_e > 0 \) partial private key queries and \( q_{H2} > 0 \) \( H2 \) queries. Assume \( A \) can break our scheme in polynomial time with probability \( \epsilon(k) \); there must be an adversary \( B \) that can solve the BDH problem with advantage at least \( \epsilon'(k) \), where \( \epsilon'(k) \geq 2\epsilon(k)/e(1 + q_e)q_{H2} \), \( e \) is the base of the natural logarithm.

Theorem 2 reduces our IBE-Signal scheme to the BDH problem. To prove this reduction, firstly, we reduce the IBE-Signal scheme to a nonidentity-based encryption scheme BasicPub. Then, we reduce the BasicPub scheme to the BDH problem. The transitivity of the reduction is noticeable.

Lemma 4. Assume \( H_i \) is a random oracle model from \( \{0, 1\}^* \) to \( G^* \) and the adversary \( A \) attacks the IBE-Signal scheme with advantage \( \epsilon(k) \) in the IND-ID-CPA game. Assume \( A \) can run at most \( q_e > 0 \) partial private key queries. There must be an adversary \( B \) that can attack BasicPub successfully with an advantage \( \epsilon(k)/e(1 + q_e) \) in the IND-ID-CPA game.

Proof. The challenger first constructs the BasicPub scheme. \( B \) attacks the BasicPub scheme by taking \( A \) as a subroutine.

4.1. Establish the BasicPub scheme.

The challenger generates public parameters of the BasicPub scheme.

\[
K_{pub} = \left( q_e, G_1, G_2, e, k, k, p, P, \text{\textbf{Y}}, Q_{ID}, H_2, H_3, H_4, \right).
\]

(27)

And keeps the key \( a \) and the partial key \( S_i \) private, where \( i = 0, 1, 2, \ldots, n \). The challenger sends \( K_{pub} \) to the adversary \( B \).

4.2. \( B \) partial private key queries.

The challenger generates public parameters of the BasicPub scheme.

\[
K_{pub} = \left( q_e, G_1, G_2, e, k, k, p, P, \text{\textbf{Y}}, Q_{ID}, H_2, H_3, H_4, \right).
\]

And keeps the key \( a \) and the partial key \( S_i \) private, where \( i = 0, 1, 2, \ldots, n \). The challenger sends \( K_{pub} \) to the adversary \( B \).

The following 4.3–4.8 steps, \( B \) simulates \( A \)’s challenger to play IND-ID-CPA game with \( A \).
4.3. Establish the IBE-Signal scheme.

B sends the IBE-Signal scheme’s public parameters
\[ K_{pub}^r = \left( q, G_1, G_2, e, k, k, P, P_i, Y_i, H_1, H_2, H_3, H_4 \right) \],
\[ H_5, H_6, \text{KDF}, g, g_1, g_2, h_1, h_2, h_3, r, rk \] \] (28)
to A. Since \( K_{pub} \) of the BasicPub scheme does not contain \( H_1 \), B needs to construct a 4-tuple \( H_1^{list} (I_D, Q_{ID}, b_j, \text{coin}) \) to simulate A’s challenger.

4.4. \( H_1 \) queries.

(i) If \( A \) queries the \( H_1 \) values of \( I_D, P \), B will respond as follows:

(ii) If \( I_D \) already exists in \( H_1^{list} \), B responds \( Q_{ID} \in G_7 \).

(iii) Else, B randomly chooses \( \text{coin} \sim R \{0, 1\} \)

(iv) and sets \( \text{Pr} [\text{coin} = 0] = \delta \), then B chooses \( b_j \sim R \{Z_q^*\} \).

(v) If \( \text{coin} = 0 \), computes \( Q_{ID} = [b_j]P \in G_7 \).

(vi) Else, \( b \) computes \( Q_{ID} = \sum_{i=0}^{n} \text{coin}_i \).

(vii) B adds \( (ID, Q_{ID}, b_j, \text{coin}) \) to \( H_1^{list} \) and responds \( Q_{ID} \) to A.

where \( \text{coin} = 0 \) represents that B thinks A will challenge \( ID^* \) in this query.

4.5. A partial private key queries – Phase I.

This step and step 4.7 can run at most \( q_E \) times in total. Assume A initiates a query on \( ID_j \) to B and each round of the query is independent.

(i) B takes \( (ID_j, Q_{ID}, b_j, \text{coin}) \) from \( H_1^{list} \).

(ii) If \( \text{coin} = 0 \), B reports an error and exits.

(iii) Else, B computes \( S_{ID_j} = [b_j] \) \( Y_i = [b_j] \) \( \prod_{i=0}^{n} S_{h_i} \). \( P = \prod_{i=0}^{n} S_{h_i} \) \( Q_{ID} \).

(iv) and sends \( \{S_{ID_j}\} \) to A as the partial private key.

4.6. challenge.

Assume A challenges \( (ID^*, M_0, M_1) \). B takes \( (ID_j, Q_{ID}, b_j, \text{coin}) \) from \( H_1^{list} \) so that \( ID_j = ID^* \).

(i) \( A \) reports an error and exits.

(ii) \( B \) computes \( S_{ID_j} = [b_j] \) \( Q_{ID} = [b_j] \) \( \prod_{i=0}^{n} S_{h_i} \).

(iii) \( = \prod_{i=0}^{n} S_{h_i} \). \( S_{ID_j} \).

(iv) and sends \( \{S_{ID_j}\} \) to A as the partial private key.

(v) \( B \) sends \( M_0, M_1 \) to A.

(vi) \( A \) initiates a query on \( ID_j \).

(vii) \( B \) computes \( b \) \( = [b_j] \) \( \prod_{i=0}^{n} S_{h_i} \).

(viii) \( B \) sends \( \text{coin} \) to A as his response.

4.7. A partial private key queries – Phase II.

Phase II is the same as Phase I. 

4.8. Guess.

A outputs \( \beta \) as his guess and sends it to B. B sends \( \beta \) to the challenger as his guess.

\[ \square \]

Assertion 1. In the above reduction process, if B does not exist, the simulation of \( B \) is complete.

From Assertion 1, \( A \)’s advantage in a simulated attack is equal to that in an actual attack, at least \( \epsilon (k) \). If A attacks the IBE-Signal scheme successfully in step 4.8, B can attack the BasicPub scheme successfully.

Since the probability of B without interruption is \((1 - \delta)^{4.5}\) steps in 4.5, 4.7, and \( \delta \) in step 4.6. The advantage \( \text{Adv}_{\text{IND-CPA}}^{\text{BasicPub}, b}(k) \) is at least \((1 - \delta)^{4.5} \cdot \delta \cdot \epsilon (k) \). Consider \( \text{Adv}_{\text{IND-CPA}}^{\text{BasicPub}, b}(k) \) is the function of \( \delta \). It can be calculated that when \( \delta = \frac{1}{(q_E + 1)} \), \( \text{Adv}_{\text{IND-CPA}}^{\text{BasicPub}, b}(k) \) reaches the maximum.

\[ \text{Adv}_{\text{IND-CPA}}^{\text{BasicPub}, b}(k) \] max \[ = \left( 1 - \frac{1}{q_E + 1} \right)^{q_E} \left( \frac{1}{q_E + 1} \cdot \epsilon (k) \right) \]
\[ > \left( \lim_{q_E \rightarrow \infty} \left( 1 - \frac{1}{q_E + 1} \right)^{q_E} \right) \cdot \epsilon (k) \]
\[ = \frac{\epsilon (k)}{\epsilon (q_E + 1)} \] (30)

Therefore, the advantage of B is at least \( \epsilon (k) / (1 + q_E) \).

Lemma 5. Assume \( H_2 \) is a random oracle model from \( G_2 \) to \( \{0, 1\}^* \). The adversary A attacks the BasicPub scheme with an advantage \( \epsilon (k) \) and A can run at most \( q_{H_2} > O \text{DH queries.} \)

There must be an adversary \( B \) can solve the BDH problem on \( G \) with an advantage \( 2 \epsilon (k)/q_{H_2} \).

Proof. B already knows \( (P, [a]P, [b]P, [c]P) \) and intends to compute \( e(P, P)^{abc} \) through the attack of A on the BasicPub scheme.

5.1. B generates public parameters of the BasicPub scheme

\[ K_{pub} = \left( q, G_1, G_2, e, k, k, P, P_i, Y_i, Q_{ID}, H_2, H_3, H_4 \right) \] \( H_5, H_6, \text{KDF}, g, g_1, g_2, h_1, h_2, h_3, r, rk \) \] (31)

Some parameters of \( K_{pub} \) are generated as follows: Let \( Q_{ID} \) be \([b]P \). B randomly chooses \( k \) as the honest CPC’s serial number and generates \( S_i (i = 0, 1, \ldots, n, i \neq k) \), then computes \( P_k = [a]P, P_i = [S_i]P (i = 0, 1, \ldots, n, i \neq k), Y_j = [\prod_{i=0}^{n} S_i]P (i = 0, 1, 2, \ldots, k - 1), Y_j = [\prod_{i=0}^{n} S_i]P (i = k, k + 1, \ldots, n) \) and \( Y = Y_n = \prod_{i=0}^{n} S_i \).

5.2. \( H_2 \) queries.

B constructs a 2-tuple \( H_2^{list}(F_1, H_2) \). A can issue \( H_2 \) queries at any time and at most \( q_{H_2} \) times.

Security and Communication Networks
If $F_j$ already exists in $H_{list}$, B responds $H_2(F_j) = H_j$.

(ii) Else, randomly chooses $H_j \leftarrow R_{[0,1]}$, B responds $H_2(F_j) = H_j$ and add $(F_j, H_j)$ into $H_{list}$.

5.3. Partial private key queries.

B generates the partial private keys $\{S_{ID_i} = \prod_{h=0, h \neq k}^{n} S_h Q_{ID_i}]_{i=0,1,\ldots,n,i \neq k}\}$, then sends them to A.

5.4. Challenge.

A outputs $M_0, M_1$ to challenge. B randomly chooses $\beta' \leftarrow R_{[0,1]}$ and sends $C^* = (U, V, rc)$ to A. Let $U$ be $[\prod_{h=0, h \neq k}^{n} S_h^{-1}] \cdot c]P$. To decrypt $C^*$, A should compute:

$$H_2(S_{ID}, U) = H_2\left(\prod_{h=0, h \neq k}^{n} S_h \cdot a \cdot Q_{ID}, \prod_{h=0, h \neq k}^{n} S_h^{-1} \cdot c \cdot P\right)$$

$$= H_2\left(\prod_{h=0, h \neq k}^{n} S_h \cdot a \cdot b \cdot P, \prod_{h=0, h \neq k}^{n} S_h^{-1} \cdot c \cdot P\right)$$

$$= H_2([ab]P, [c]P)$$

$$= H_2(P, P)^{abc}. \quad (32)$$

A issues the $H_2$ queries before decrypting $C^*$ and the BDH problem’s solution will be embedded in $H_{list}$.

5.5. Guess.

A outputs $\beta' \in [0,1]$. Meanwhile, B randomly chooses $(F_j, H_j)$ from $H_{list}$ and let $F_j$ be the solution to the BDH problem.

Assertion 2. In the above simulation process, the simulation of B is complete.

Assertion 3. Let $\Lambda$ denotes the event: in the above simulation, A has issued the query $H_2(e(S_{ID}, U))$. Then, $Pr[\Lambda] \geq 2\varepsilon$. 

---

Table 3: Computational overhead of operations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
<th>In $T_M$</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_M$</td>
<td>Modular multiplication</td>
<td>$1.00 T_M$</td>
<td>$\approx 0.13$</td>
</tr>
<tr>
<td>$T_l$</td>
<td>Modular inversion</td>
<td>$11.60 T_M$</td>
<td>$\approx 1.51$</td>
</tr>
<tr>
<td>$T_A$</td>
<td>Two elliptic curve points addition</td>
<td>$0.12 T_M$</td>
<td>$\approx 0.02$</td>
</tr>
<tr>
<td>$T_S$</td>
<td>Elliptic curve scalar point multiplication</td>
<td>$29.00 T_M$</td>
<td>$\approx 3.77$</td>
</tr>
<tr>
<td>$T_E$</td>
<td>Exponentiation</td>
<td>$240.00 T_M$</td>
<td>$\approx 31.2$</td>
</tr>
<tr>
<td>$T_{H1}$</td>
<td>One-to-one hash mapping</td>
<td>$29.00 T_M$</td>
<td>$\approx 3.77$</td>
</tr>
<tr>
<td>$T_{H2}$</td>
<td>Two-to-one hash mapping</td>
<td>$120.23 T_M$</td>
<td>$\approx 15.63$</td>
</tr>
<tr>
<td>$T_p$</td>
<td>Bilinear pairing</td>
<td>$87.00 T_M$</td>
<td>$\approx 11.31$</td>
</tr>
<tr>
<td>$T_{AES-128}$</td>
<td>Encryption or decryption of the 1 KB size message using AES-128 at CTR mode</td>
<td>$26.77 T_M$</td>
<td>$\approx 3.48$</td>
</tr>
</tbody>
</table>

---
Table 4: Computational overhead of the subalgorithms in our scheme IBE-Signal for \( n = 10 \), the number of the CPCs and \( nu = 1,000,000,000 \), and the number of users.

<table>
<thead>
<tr>
<th>Phases</th>
<th>Operations</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGC setup</td>
<td>( T_S )</td>
<td>3.77</td>
</tr>
<tr>
<td>CPC setup</td>
<td>( n(n-1)/2T_P + nT_S )</td>
<td>546.65</td>
</tr>
<tr>
<td>Key authority setup</td>
<td>( T_S )</td>
<td>3.77</td>
</tr>
<tr>
<td>Encryption RootInput generation</td>
<td>( 2T_{Ha} + T_S + T_P + T_E )</td>
<td>53.82</td>
</tr>
<tr>
<td><strong>Encryption key generation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encryption revocation key generation</td>
<td>( 3T_{Ha} + T_P + T_E + T_A )</td>
<td>115.47</td>
</tr>
<tr>
<td>Sign</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sender's private key generation</td>
<td>( T_I + (2n + 5) T_P + T_A )</td>
<td>276.72</td>
</tr>
<tr>
<td>Signature generation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiver's private key generation</td>
<td>( (n + 2)T_{Ha} + (n + 2) T_P )</td>
<td>15.08</td>
</tr>
<tr>
<td>Decryption RootInput generation</td>
<td>( 3T_{Ha} + 3T_M + T_HB )</td>
<td>657.95</td>
</tr>
<tr>
<td><strong>Decryption key generation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decryption key updates generation</td>
<td>( [\log_{2} nu + 1] (7T_S + 3T_M) + T_{Ha} )</td>
<td>807.17</td>
</tr>
<tr>
<td>Revocation key selection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decryption revocation key generation</td>
<td>( 2T_M + 2T_I + 4T_P )</td>
<td>130.32</td>
</tr>
<tr>
<td>Decrypt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revocation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SESSION total**

\( (12 + 6[\log_{2} nu]) T_M + 4T_I + T_A + (34 + 5n + 14[\log_{2} nu]) T_S + + (6T_E + (2n + 14) T_{Ha} + 4T_{Hb} + (n(n + 3)/2 + 14) T_P + 2T_{AES-128} \\n\)

Verify

\( 2T_S + T_{Ha} + T_{Hb} \) or \( T_{Ha} \)

26.94

3.48

23.17

3.48

**CONNECTION total**

\( 3T_S + 2T_{Ha} + 2T_{Hb} + 2T_{AES-128} \)

57.07

**Total**

\( (12 + 6[\log_{2} nu]) T_M + 4T_I + T_A + (37 + 5n + 14[\log_{2} nu]) T_S + + (6T_E + (2n + 16) T_{Ha} + 6T_{Hb} + (n(n + 3)/2 + 14) T_P + 4T_{AES-128} \)

3263.52
Figure 15: The computational overhead of each subalgorithm of the IBE-Signal scheme.

Figure 16: The computational overhead of Revocation Private Keys Generation and Revocation Key Updates Generation.
Table 5: The computational overhead of our scheme and other schemes.

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Setup and key extraction</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kumar and Chand [17]</td>
<td>$(4n + 6)(T_P + T_S) + T_{H_a}$</td>
<td>$= 60.32n + 94.25$</td>
</tr>
<tr>
<td>Chen et al. [41]</td>
<td>$(3 + n)T_S + (2 + n)T_A$</td>
<td>$= 3.79n + 11.35$</td>
</tr>
<tr>
<td>Chen et al. [42]</td>
<td>$(4n + 7)(T_S + T_A)$</td>
<td>$= 15.16n + 26.53$</td>
</tr>
<tr>
<td>Li et al. [43]</td>
<td>$(2 + n)T_S + nT_A + T_P$</td>
<td>$= 3.79n + 18.85$</td>
</tr>
<tr>
<td>Our scheme</td>
<td>$T_I + (3n + 7)T_S + (n + 2)T_{H_a} + (n^2 + n + 4)/2T_P$</td>
<td>$= 5.655n^2 + 20.735n + 58.06$</td>
</tr>
</tbody>
</table>

Figure 17: Comparison of the computational overhead with other schemes.

Table 6: Comparison of the security features between our scheme and other schemes.

<table>
<thead>
<tr>
<th>Schemes</th>
<th>RKEP</th>
<th>RSKIP</th>
<th>RUSP</th>
<th>SMA</th>
<th>SPFS</th>
<th>SPCS</th>
<th>SKR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kumar and Chand [17]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Chen et al. [41]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Chen et al. [42]</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Li et al. [43]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Boldyreva et al. [18]</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Our scheme</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Proof. Obviously, \( \Pr[\beta' = \beta] = 1/2 \) We know \( |\Pr[\beta' = \beta] - 1/2| \geq \varepsilon \). So,

\[
\Pr[\beta' = \beta] = \Pr[\beta' = \beta|\Lambda] \Pr[\Lambda] + \Pr[\beta' = \beta|\Lambda] \Pr[\Lambda] - \frac{1}{2} \Pr[\Lambda] + \Pr[\Lambda] \\
\leq \Pr[\beta' = \beta|\Lambda] \Pr[\Lambda] + \Pr[\Lambda] - \frac{1}{2} \Pr[\Lambda] + \Pr[\Lambda] \\
= \frac{1}{2} (1 - \Pr[\Lambda]) + \Pr[\Lambda] - \frac{1}{2} \Pr[\Lambda] + \Pr[\Lambda] \\
= \frac{1}{2} (1 - \Pr[\Lambda]) + \Pr[\Lambda] - \frac{1}{2} \Pr[\Lambda] + \Pr[\Lambda] \\
\epsilon \leq |\Pr[\beta' = \beta] - \frac{1}{2}| \leq \frac{1}{2} \Pr[\Lambda] \\
\Pr[\Lambda] \geq 2\epsilon. \tag{33}
\]

From Assertion 2 and Assertion 3, B solves the BDH problem with the probability at least \( 2\epsilon/|q_{H}| \).

From Lemma 4, there is an adversary \( B \) that attacks the BasicPub scheme with the advantage of at least \( \epsilon_j = \epsilon(1 + q_{p}) \). From Lemma 5, there is an adversary \( B \) that solves the BDH problem on \( G \) with the advantage of at least \( 2\epsilon/|q_{H_{1}}| = 2\epsilon(1 + q_{E})/|q_{H_{1}}| \). \( \square \)

6. Performance Evaluation

We mainly evaluated the performance of our scheme from the computational overhead and the security features. To facilitate the calculation of overhead, we implemented it by calculating and combining cryptographic operations, which is also adopted in [17, 36–39]. Moreover, the implementation of the IBE-Signal scheme was based on OpenSSL v1.1.1f and the Type-A curve of the PBC library and was performed on a Dell Inspiron 7580 with Intel Core i7-8565U CPU (1.80 GHz) and 16 GB memory, running Ubuntu 64-bit Linux (v20.04.4 LTS). For the pairing-based scheme, to achieve the 1,024-bit RSA level security, we used the Tate pairing defined over the supersingular elliptic curve \( E/F_p \): \( y^2 = x^3 + x \) with embedding degree 2, where \( p \) is a 160-bit Solinas prime \( q = 2^{159} + 2^{17} + 1 \) and \( p \) a 512-bit prime satisfying \( p + 1 = 12q_p \). For the ECC-based schemes, to achieve the same security level, we employed the ECC group on Kobritz elliptic curve \( y^2 = x^3 + ax^2 + b \) defined on \( F_{2^{512}} \) with \( a = 1 \) and \( b \) a 163-bit random prime.

6.1. The Computational Overhead of Our Scheme. Table 3 shows the time taken for the cryptographic operations involved in this paper. We define the overhead of modular multiplication as \( T_M \), the overhead of modular inversion as \( T_I \), the overhead of two elliptic curve points addition as \( T_A \), the overhead of elliptic curve scalar point multiplication as \( T_S \), the overhead of exponentiation as \( T_E \), the overhead of one-to-one hash mapping as \( T_{H_{\text{A}}} \), the overhead of two-to-one hash mapping as \( T_{H_{\text{F}}} \), the overhead of bilinear pairing as \( T_P \), and the overhead of encryption or decryption of the 1 KB size message using AES-128 at CTR mode as \( T_{\text{AES-128}} \). From [17], we have obtained \( T_I = 11.60T_M \) and \( T_{H_{\text{A}}} = 29.00T_M \). Then, we calculated the overheads of \( T_M = 0.13\text{ms} \), \( T_{H_{\text{F}}} = 15.63\text{ms} \), and \( T_{\text{AES-128}} = 3.48\text{ms} \) separately. In [40], \( T_A = 0.12T_M \), \( T_S = 29.00T_M \), and \( T_E = 240.00T_M \) have been given. It has been given in [39] that the overhead of bilinear pairing \( T_P = 3.00T_M = 87.00T_M \).

As shown in Table 4, we calculated the number of operations and the computational overhead of each subalgorithm of the IBE-Signal scheme according to Table 3. Among them, the CPCs are selected as ten, and the number of users is selected as 1 billion. The overhead of the SESSION part of our scheme is 3206.45 ms. However, the SESSION part only runs when the two opposing parties communicate for the first time, so a computational overhead of 3 seconds is acceptable. After the SESSION part is established, the CONNECTION part takes only 57.07 ms. Further, if the communication parties are not in the first message in each round of session, the computational overhead can be reduced to 14.5 ms.

The number of CPCs \( n \) is essential in determining the system’s computational overhead. Theoretically, the more the number of the CPCs, the lower the risk of the key escrow problem, but the computational overhead also increases. In Figure 15, we show the relationship between the computational overhead of each subalgorithm and \( n \) from 1 to 10. Among them, the computational overhead of three subalgorithms CPC Setup, Sender’s Private Key Generation, and Receiver’s Private Key Generation increases as \( n \) increases. In addition to the number of the CPCs, the computational overhead of the two subalgorithms Revocation Private Keys Generation and Revocation Key Updates Generation is related to the number of users \( m_n \). We give the overhead trend as \( m_n \) grows in Figure 16, where \( m_n \) grows from 1 million to 1 billion. Considering the monthly active users in Table 1, we set the number of users to 1 billion. Furthermore, the algorithm’s time complexity is \( O(\log N) \) due to a binary tree structure. This results in a computational overhead of only about 800 ms, even with a billion users.

6.2. Comparison of the Computational Overhead with Other Schemes. In Table 5 and Figure 17, we compare the computational overhead between our scheme and other schemes. Moreover, the comparison mainly focuses on the two stages of Setup and Key Extraction. As shown in Figure 17, the SESSION part overhead of our scheme is lower than the Kumar et al. scheme [17] when the number of the CPCs is less than 8. Even if there are less than 8 CPCs, they can already provide sufficient private key distribution protection. Moreover, the CONNECTION part of our scheme is a
constant function for \( n \). Our scheme runs the \textit{CONNECTION} part in most cases, so our scheme is still competitive in computational overhead.

6.3. Comparison of the Security Features with Other Schemes. Table 6 compares our scheme with other schemes in terms of security features. We define Resilient to the Key Escrow Problem as RKEP, Resilient to the Secure Key Issuing Problem as RSKIP, Resilient to the User Slandering Problem as RUSP, Support Mutual Authentication as SMA, Support Perfect Forward Secrecy as SPFS, Support Post-compromise Security as SPCS, and Support Key Revocation as SKR. The KGC cannot directly generate private keys for users in our scheme, so the scheme has the RKEP feature. The blind signature enables the KGC or the CPCs to issue the private key without seeing the actual information sent by the user, which provides the RSKIP feature for the scheme. The CPCs have a master key slice, which spreads the risk of private key distribution and provides the RUSP feature. Furthermore, we provide the SMA feature by introducing IBS. Since our \textit{CONNECTION} part is based on the Double Ratchet algorithm, we provide both the SPFS feature and the SPCS feature. Finally, we introduce the Revocable IBE to provide the SKR feature.

7. Conclusion

We have proposed the IBE-Signal scheme to reshape the Signal into a MITM-attack-resistant protocol. Our scheme provides more security features (such as Perfect Forward Secrecy, Post-compromise Security, and Key Revocation) than other schemes. Furthermore, experiments show that our scheme has less computational overhead when the number of the CPCs is less than 8. Moreover, we proved that our scheme is IND-ID-CPA secure under the random oracle model, even if only one CPC is credible. We increased the attackers’ capabilities in the proofs: some partial private keys of ID* can still be queried in the challenge phase. In the future, we will transform the IND-ID-CPA secure scheme into IND-ID-CCA secure scheme using the Fujisaki-Okamoto method [44].

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no known conflicts of interest or personal relationships that could have appeared to influence the work reported in this study.

Acknowledgments

This work was supported by Research Funds for NSD Construction, University of International Relations (2020GA04) and National Natural Science Foundation of China (62102113).

References


24 Security and Communication Networks


