A Non-Reciprocal Key Agreement Scheme Based on SM9 Algorithm

Xianglong Dai
School of Cyber Science and Engineering, Southeast University
Nanjing, China
220224938@seu.edu.cn

Lei Mao
School of Information and Electronics Engineering, Jiangsu Vocational Institute of Architectural Technology
Xuzhou, China

Su Shu
Shenzhen Sunline Tech Co., Ltd. (Chengdu Branch)
Chengdu, China

Ge Wu*
School of Cyber Science and Engineering, Southeast University
Nanjing, China
gewu@seu.edu.cn

ABSTRACT

Key agreement protocols are essential in practical encryption scenarios since a session key is generated from the public-key setting and the session key will be applied in later communications. Identity-based cryptography relieved the heavy burden in public key infrastructure. However, similar to the traditional public-key setting, most identity-based key agreement protocols are suitable for two equal parties. In the client-server model or some environments that requires timely response communications, a more powerful server is normally established and the client is comparatively resource-constrained. Making the protocols more efficient for these scenarios can improve the authentication speed in many applications. In this paper, we apply and improve the online/offline transformation for the identity-based key agreement protocol, namely SM9 algorithm. Based on this nation cryptographic standard, we propose a non-reciprocal fast key agreement scheme, which can effectively solve the aforementioned shortcomings in traditional protocols.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

SM9 key agreement, Online/Offline transformation, Non-reciprocal pairing operation

ACM Reference Format:
Xianglong Dai, Lei Mao, Su Shu, and Ge Wu. 2018. A Non-Reciprocal Key Agreement Scheme Based on SM9 Algorithm. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

*Corresponding Author (Ge Wu: gewu@seu.edu.cn).

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Traditional public key cryptography (PKC) relies on a third trusted party, called certificate authority (CA) to verify user's identity and issue the corresponding certificate. The public key certificates are used to bind public keys to user identities. However, the certificate management process can be cumbersome and inefficient, especially for large-scale networks with a high volume of users. In 1984, Shamir proposed identity-based cryptography(IBC) as a solution to the challenging issues of certificate system storage and management under the traditional public key system. This paradigm is effective in resisting key replacement attacks. IBC solves the certificate problem of traditional public key cryptography by generating the user's private key from a predefined function using the relevant string that identifies the entity. The private key is generated by the Public Key Generator (PKG) and securely transmitted to the user. The private key contains user information, making it impossible for the key holder to deny their actions.

As part of the IBC, identity-based authenticated key agreement (IBAKA) is an important cryptographic primitive that allows two parties to establish a shared session key over an insecure channel, while authenticating each other's identities. IBAKA is a key building block for many secure applications, such as secure messaging, e-commerce, and mobile payments. The SM9 algorithm is the national cryptographic standard with respect to the identity-based algorithm. It was developed and published by the Chinese government. This standard includes algorithms for encryption, signature, and key agreement. In particular, The the key agreement algorithm is normally specified as SM9-KA.

However, most practical IBAKA schemes, including SM9-KA, are based on pairing groups, which require pairing operations and scalar multiplication operations, leading to expensive computational overhead. In addition, the key agreement phase is normally conducted between two equal parties. However, in the scenarios where the two parties have different resources (like in client-server

model), or a timely response is required (like in high-speed movement environment), these algorithms cannot effectively fit the aforementioned practical applications. Therefore, how to improve these unfavorable algorithms more efficient is worthy studying.

1.1 Motivations

There is a scenario involving high-mobility communication, where resource constrained devices, such as drones and train-mounted communication units, require secure communication with a central base station. These lightweight devices face limitations in processing power and memory, rendering them incapable of executing computationally expensive cryptographic operations. Additionally, the high-speed movement introduces transient communication windows with the corresponding base station, making secure and time-sensitive data exchange a significant challenge. Conversely, the base station possesses superior computational capabilities and can tolerate longer processing delays. This asymmetry in resource availability necessitates a cryptographic scheme that minimizes the computational burden on lightweight devices while leveraging the base station's processing power to ensure secure and efficient communication. Therefore, exchanging information quickly and reducing the complexity of operation of the lightweight devices under these circumstances has become a challenging task.

The online/offline transformation is a common efficiency improvement in identity-based cryptography that can play an important role in the scenario described above. However, there is no online/offline IBAKA protocol to the best of our knowledge. In addition, the online/offline transformation does not deal with issues of pairing operations in client side. The drawback of the common online/offline transformation on IBAKA and how to improve this transformation to make it more adaptable in IBAKA remains unknown.

SM9-KA as a key agreement algorithm for government mandated use also has the above drawbacks. In this paper, we address the limitations of the current SM9-KA to solve its problems in the above challenges.

1.2 Our Contributions

Through our research, we propose a non-reciprocal fast key agreement scheme based on SM9 in this paper. The main contributions of this paper are as follows:

- We have improved and applied Online/Offline transformation to SM9-KA for the first time.
- On this basis, we show how to deal with the pairing operations in a non-reciprocal way to improve the SM9 algorithm more practical for lightweight client-to-server key agreement scenarios.
- Security proofs and theoretical analyses are provided. In addition, the efficiency comparisons between our scheme and SM9-KA demonstrates that our scheme is more suitable for the aforementioned timely response scenarios.

1.3 Related Work

This subsection provides a brief description of the related work on online/offline algorithms and SM9.

The core idea of the online/offline algorithm is to divide the algorithm into two phases, namely the offline phase and the online phase. The operations during the offline phase is independent of the corresponding input of the algorithm and hence can be computed and the results can be stored in advance. During the online phase, the pre-computed and stored results together with the input of the algorithm are involved to perform comparatively lightweight operations. The algorithm was first proposed in 1990 by Even et al. in [8] and applied to signature algorithms as a solution to the time-consuming problem of signature generation. Since then, online/offline signature algorithms have been the subject of some research by experts. The main research direction focuses on efficiency improvement[9, 11], online/offline signature algorithms based on various cryptographic components[22], and the way of applying them in various real-world scenarios[1, 13, 21]. The scheme proposed by Liu et al[16] became one of the ISO/IEC standards. In the scheme [10] proposed by Guo et al. in 2008, online/offline algorithm was extended to encryption schemes for the first time. Since then, a series of researches have also been conducted on online/offline algorithm for encryption schemes [6, 15, 17–19]. Online/offline algorithm has also been used in key encapsulation[6] and attribute encryption[7], and in the SM9 algorithm[14]. However, in our research, the online/offline algorithm has not yet been used in key agreement schemes so far, and the reasons for this situation and our improvements for its adaptation in key agreement are mentioned in the later of this paper.

Turn to the research on SM9 algorithms. Cheng et al. in [4] summarised and proved the security of SM9 encryption, signature and key agreement. There have also been a number of studies, mainly by Chinese researchers, on improving various SM9 algorithms in terms of efficiency and practical application[14, 20]. The improvement of SM9-KA has also been studied by some scholars and postgraduate students[12]. But unfortunately, the SM9-KA has the unavoidable problem of pairing, and scientists have not come up with a very effective solution. In this paper, we propose a unique solution to the pairing problem, focusing on improving the efficiency of one side and providing a new optimisation scheme for the pairing operation of SM9-KA.

1.4 Organization

Some related basic tools and the original SM9-KA algorithm are described in Chapter 2. The security model, the adversary model are given in Chapter 3. Our improvement, and the complete scheme are described in detail in Chapter 4. Formal security proofs are given in Chapter 5. Some comparisons between our scheme and the original SM9-KA are given in Chapter 6. Finally, a conclusion is given in Chapter 7.

2 PRELIMINARIES

2.1 Pairing and Related Hardness Problems

Bilinear pairing: A pairing operation is defined as a bilinear map:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$$

Note that $\mathbb{G}_1$ and $\mathbb{G}_2$ are usually two additive cyclic groups of prime order p. $\mathbb{G}_T$ is a multiplicative cyclic group. This map should meet the following three conditions:

- Bilinearity: For any $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b \in \mathbb{Z}_p^*$, we have $e([\alpha]P, [\beta]Q) = e(P, Q)^{\alpha\beta}$.
- Non-degeneracy: For any generator $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, we have $e(P, Q) \neq 1 \in \mathbb{G}_T$.
- Computability: For any $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b \in \mathbb{Z}_p^*$, there exists a probabilistic polynomial time (PPT) algorithm to compute $e([\alpha]P, [\beta]Q)$.

DBIDH Hard Problem [3]: Given $x, y, r \in_R \mathbb{Z}_p^*$, for $i, j \in \{1, 2\}$, distinguishing: $(P_1, P_2, [x]P_i, [y]P_j, e(P_1, P_2)^{y/x})$ with $(P_1, P_2, [x]P_i, [y]P_j, e(P_1, P_2)^r)$ is hard.

Gap$-\tau-$BCAA1 Hard Problem[5]: When given $x \in_R \mathbb{Z}_p^*$, and if we have τ values: $(P_1, P_2, [x]P_i, h_0, (h_1, [\frac{x}{h_1+x}]P_j), ..., (h_\tau, [\frac{x}{h_\tau+x}]P_j))$, in this $i, j \in \{1, 2\}$ and values $h_i \in_R \mathbb{Z}_p^*$ $(0 \leq i \leq \tau)$ different from each other, given a *DBIDH* oracle, calculating $e(P_1, P_2)^{x/(h_0+x)}$ is tough.

$\psi-$BDH Hard Problem[5]: For $x, y \in_R \mathbb{Z}_p^*$, given $(P_i, P_j, [x]P_j, [y]P_j)$ that $i, j \in \{1, 2\}$ and $i \neq j$, computing $e(P_1, P_2)^{xy}$ is hard if there does not exist a mapping: $\phi : \mathbb{G}_j \to \mathbb{G}_i$ that $\phi(P_j) = P_i$ is efficiently computable.

2.2 SM9 Key Agreement Scheme

An SM9 Key Agreement (SM9-KA) is composed of four parts: **Setup**, **PrivateKeyExtract**, **MessageExchange**, **SessionKeyGeneration**. The following is a detailed description of the operation of each part:

Setup(1^λ): Inputting a secure parameter λ, PKG will generate the system parameters in the following way:

(1) Choose three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Randomly choose generators from groups: $P_1 \in \mathbb{G}_1, P_2 \in \mathbb{G}_2$.
(2) Randomly choose the master private key $s \in \mathbb{Z}_p^*$ and compute the related master public key $P_{pub} = [s]P_1$.
(3) Calculate the median value in advance: $v = e(P_{pub}, P_2)$.
(4) Decide a hash function H_1 and an additional bit *hid*.
(5) Output the master public key and public parameters:$Params =< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, P_{pub}, v, H_1, hid >$, and the master private key is $msk = s$.

PrivateKeyExtract$(Params, msk, ID_i)$**:** When PKG receives a party U_i's identify ID_i, it uses its master private key *msk* to generate the party's private key:

$$D_i = [\frac{s}{s + H_1(ID_i||hid, p)}]P_2$$

Note that if $s + H_1(ID_i||hid, p) \equiv 0 \bmod p$, the PKG should react the **Setup** step and change a new master secret key. The PKG then sends the key to the party over a secure channel.

MessageExchange: If a party U_A wants to communicate with the party U_B, the U_A and U_B will generate $r_A, r_B \in \mathbb{Z}_p^*$ locally successively. Then they compute and exchange the messages as follows. Note that the S_A, S_B shown below are the optional parts used to confirm the session key.

$$A \to B : R_A = [r_A]([H_1(ID_B||hid, p)]P_1 + P_{pub})$$
$$B \to A : R_B = [r_B]([H_1(ID_A||hid, p)]P_1 + P_{pub})\{, S_B\}$$
$$\{A \to B : S_A\}$$

SessionKeyGeneration: When the two parties U_A and U_B have acknowledged receipt of the message, they can generate the session key locally. For U_A, it should compute the intermediate process as follows:

$$\begin{cases} g_1 = e(R_B, D_A) \\ g_2 = v^{r_A} \\ g_3 = g_1^{r_A} \end{cases}$$

And then it can use the key derived function (KDF) to generate the session key:

$$SK_A = KDF(ID_A, ID_B, R_A, R_B, g_1, g_2, g_3, l)$$

While for another party U_B, it should compute the intermediate process as follows:

$$\begin{cases} g_1' = v^{r_B} \\ g_2' = e(R_A, D_B) \\ g_3' = g_2'^{r_B} \end{cases}$$

Then it can generate the same session key as U_A using the key derived function:

$$SK_B = KDF(ID_A, ID_B, R_A, R_B, g_1', g_2', g_3', l)$$

Upon completion of the steps above, the parties U_A and U_B can hold the same session key.

3 DEFINITION AND SECURITY MODEL

In our scheme, we apply and improve the online/offline transformation algorithm to achieve timely responsibility. The traditional online/offline algorithm divides the message sending process into online phase and offline phase. Whereas in our scheme, the offline phase can be computed once and used multiple times. To give the server the ability to calculate offline in advance, we transfer the offline calculation values from the client to the server over a secure channel in advance. This can significantly speed up the calculation process on both the client and server sides. Therefore, our IBKA scheme definition and security model are slightly different from the traditional IBKA scheme, which will be described in detail below.

3.1 Definition of the Scheme

A general IBAKA protocol is mainly composed of four algorithms: initialization algorithm **Setup**, private key extract algorithm **PrivateKeyExtract**, message exchange algorithm **MessageExchange** and session key generation algorithm **SessionKeyGeneration**. In our scheme, the client should calculate the values in advance and send them to the server over a secure channel so that the server can calculate them offline. So in our non-reciprocal fast key agreement scheme based on SM9, there are five algorithms: **Setup, PrivateKeyExtract, OfflineTransfer, MessageExchange, SessionKeyGeneration**. Each algorithm is described below:

- $Setup(1^\lambda)$: Initialization algorithm. Upon input the secure parameter λ ,the PKG outputs the system parameters $Params$. The algorithm is defined as follows:

$$(Params, msk, mpk) \leftarrow Setup(1^\lambda)$$

- $PrivateKeyExtract(Params, msk, ID_i)$: Private key extraction algorithm. Upon input the system parameters $Params$ and the party's identify ID_i, the PKG outputs the party's private key D_i. The algorithm is defined as follows:

$$D_i \leftarrow PrivateKeyExtract(Params, msk, ID_i)$$

- $OfflineTransfer$: Offline algorithm. The client should calculate the values in advance and send them to the server over a secure channel. And then the server can prepare the message offline. The algorithm is defined as follows:

$$m_{off-C} \leftarrow Offline(D_C)$$
$$m_{off-S} \leftarrow Offline(m_{off-C}, ID_C)$$

- $MessageExchange$: Message exchange algorithm. This algorithm acts between the parties U_C and U_S who want to communicate with each other. They compute and exchange the messages using established processes. The algorithm is defined as follows:

$$C \rightarrow S : m_C \leftarrow MessageExchange_C$$
$$S \rightarrow C : m_S \leftarrow MessageExchange_S$$

- $SessionKeyGeneration$: Session key generation algorithm. When the two parties mentioned above got the exchanged message, they should generate the same session key locally. The algorithm is defined as follows:

$$SK_C \leftarrow SessionKeyGeneration_A(m_S)$$
$$SK_S \leftarrow SessionKeyGeneration_B(m_{off-C}, m_C, D_S)$$

3.2 Security Model

The modified Bellare-Rogaway authenticated key agreement model (mBR)[2] has been widely used in the proofs of various authenticated key agreement schemes. The model was used in the proof of SM9-KA by Cheng et al. in 2019 [5]. In this paper we have adapted some of the processes of the model to make it more adaptable to our scenario.

In mBR, processing a message exchange action is considered as an oracle $\prod_{i,j}^t$. This oracle means the t's demonstration i exchanges message with j. $\prod_{i,j}^t$ acts the algorithm mentioned above and outputs the values: $\prod(1^k, i, j, SK_i, PK_i, PK_j, tran_{i,j}^t, r_{i,j}^t, x) = (m, \delta_{i,j}^t, \sigma_{i,j}^t, j)$. $r_{i,j}^t$ is considered as the random value of the agreement. x is the input value and m is the output value. $\delta_{i,j}^t$ represents the decision and $\sigma_{i,j}^t$ represents the session key. SK and PK are the private and public key of the party. The adversary has access to the oracle through the issuing of queries.

The security of the protocol is defined as a two-phase game between the adversary $\mathcal{A}$ and the challenger C. The process of the game is acted as C simulates the executions of the protocol by answering $\mathcal{A}$'s queries accessing the oracle. In this paper, we slightly changes the phase of the model. In the initial phase, the challenger C should first choose a party as the sender and generate the values in OfflineTransfer step. Then C give all of the appointed values to the adversary $\mathcal{A}$. Note that we assume that the sender's transmit value changes every round in this phase, in order to give the adversary the opportunity to learn. After that, the adversary can start its queries.

In the initial phase, is permitted to submit the following queries they wish at any point in time, and in whatever sequence they see fit:

- $Send(\prod_{i,j}^t, x)$. The challenger C activates oracle $\prod_{i,j}^t$ upon receipt of the message x. It executes the protocol and returns the message m or makes a decision about the acceptance or rejection of the session. If there is no such oracle $\prod_{i,j}^t$, the challenger should create an oracle defined as the sender if $x = \lambda$ or a receiver otherwise. Note that we limits $i \neq j$. That is to say, the sender and the receiver cannot be the same party.
- $Reveal(\prod_{i,j}^t)$. Return $\perp$ if the session is not accepted, or return the session key otherwise.
- $Corrupt(i)$. The challenger gives the adversary the party i's private key.

When the adversary decided to finish the first phase, it will start the second phase. The adversary will choose a fresh oracle $\prod_{i,j}^t$ and issue the $Test$ query. The fresh oracle and $Test$ query definitions are given below:

Definition 1. *We call an oracle a fresh oracle if it satisfies the following conditions.*

(1) $\prod_{i,j}^t$ *is accepted.*
(2) $\prod_{i,j}^t$ *is unopened.*
(3) *Party* $j \neq i$ *is not corrupted.*
(4) *The session* $\prod_{j,i}^w$ *which has the same session id with* $\prod_{i,j}^t$ *is not revealed.*

- $Test(\prod_{i,j}^t)$. The adversary chooses a fresh oracle $\prod_{i,j}^t$. Then the challenger randomly chooses $b \in \{0, 1\}$. If $b = 0$ the challenger returns the session key, otherwise it returns a random value.

After this phase, the adversary can continue issuing the queries of the first phase, other than revealing the oracle chosen in $Test$ stage and the oracle which matches it if the oracle exists without corrupting j. (Note that the definition of the match of the oracle is combined with e the concatenation of the messages) At last, the adversary should give a guess b' for b. If $b' = b$, we say that the adversary wins. The adversary's advantage is defined as follows:

$$ADV_{\mathcal{A}}(k) = |2Pr[b' = b] - 1|$$

Thus, the definition of our IBAKA protocol would be:

Definition 2. *If a IBAKA protocol satisfies the following conditions, it is a secure IBAKA protocol.*

(1) *The oracles* $\prod_{i,j}^t$ *and* $\prod_{j,i}^w$ *can always retain the same session key when operating in the presence of a benevolent adversary, which transmits messages faithfully. The distribution of the key is homogeneous throughout the entire session key space.*
(2) *For any polynomial time adversary* $\mathcal{A}$, $ADV_{\mathcal{A}}(k)$ *is negligible.*

If the IBAKA protocol can be proven secure, it enables implicit key authentication interactions as well as general security properties: known session secret key security, security resilience to secret key disclosure, and unknown secret key sharing resilience.

Note that there are two types of party in our scheme: the sender and the receiver. The adversary's ability of breaking the protocol from the receiver side is as above. On the sender's side, the adversary certainly does not have access to its long-term private key, and the randomness of information guarantees this.

And then we consider forward security. In order to guarantee the forward security, it is essential to ensure that even if a party's long-term secret key is subsequently compromised, the security of the previous session key is not compromised. It is defined as follows:

Definition 3. *A IBAKA protocol is forward secure if it satisfies: if any PPT ability adversary can win by only negligible advantage in any case where the adversary chooses the unrevealed oracle* $\prod_{j,i}^{w}$ *possessing the corresponding unrevealed oracle* $\prod_{i,j}^{t}$ *as a challenge, and only one of i, j is corruptible. The protocol is perfectly forward secure if the condition is changed to both i, j being corruptible. The protocol is said to satisfy master key forward security if the master private key can be made public during the game. Where the corruption of the long-term secret key as well as the leakage of the master key may occur at any time during the game.*

4 NON-RECIPROCAL KEY AGREEMENT SCHEME BASED ON SM9

4.1 Online/Offline Transformation and Improvement on SM9-AK

In the paper [15], Lai proposed a way to transform a common structured identity-based encryption algorithm into an identity-based online/offline encryption (IBOOE) algorithm. It is notable that the IBE comprises a substantial portion designated as the ID Header. It is defined as $(g_1 g^{ID})^s$. The online/offline algorithm aims to separate the ID header into offline part and online part. The encryptor can compute some complex elliptic curves or exponentiation operations offline without knowing the target's ID. Once the encryptor got the target's ID, it can just compute simple operations online and send the encrypted data to the recipient. A good way to split the ID header is:

$$\begin{cases} C_{off} & = (g_1 g^{w})^s \\ C_{on} & = s(ID - w) \bmod p \end{cases}$$

The encryptor can send the two values instead of online computing the ID header. The decryptor can easily reduct the ID header by this way:

$$Hdr = C_{off}^{C_{on}} = (g_1 g^{w})^s \cdot g^{s(ID-w)} = (g_1 g^{ID})^s$$

In this way, the complex operations of the encrypted party can be performed offline, while the online phase only requires certain additive and multiplicative operations on the group, which greatly speeds up the encryption.

Notice that there is a part similar to ID header in the SM9-KA and that this part is the main operation for performing the authenticated key agreement:

$$R_i = [r_i]([ID_i]P_1 + P_{pub})$$

It is true that the above online/offline transformation can be used in SM9-KA to achieve the effect of calculating in advance without both users knowing the identity ID of the other party, and then quickly calculating and sending the identity of the other party after it has been obtained. Anyone can use the online/offline transformation on SM9-KA to separate this part like:

$$\begin{cases} C_{off} & = [r_i]([w]P_1 + P_{pub}) \\ C_{on} & = r_i(ID - w) \bmod p \end{cases}$$

Then after exchanging the two messages, anyone can compute and use this part as usual:

$$R_i = C_{off} + [C_{on}]P_1 = [r_i]([ID_i]P_1 + P_{pub})$$

We tentatively call in our paper the SM9-KA algorithm generated by the above transformation SM9 online/offline key agreement scheme (SM9-OO-KA).

However, such an approach has limited improvement in computation speed because the separated offline values carry the temporary secret key r_i that the user wants to agree: $C_1 = [r_i]([w]P_1 + P_{pub})$. The r_i in the above equation cannot be re-used each time the key agreement protocol is performed to prevent replay attacks by malicious adversaries. It means that if the key agreement is to be performed using the online/offline approach, the user must compute and store a sufficient amount of C_{off} during the offline phase to operate with multiple users, which is not only time but also storage consuming.

After comparison, it can be seen that the difference between key agreement and encryption is that the random value s of the identity header in the encryption algorithm must be involved in the operation in the encrypting party, which has to perform the power operation to hide and then compute with the message it wants to encrypt in order to obtain the encrypted effect. It means that the random value s cannot be determined in the online phase in encryption. Whereas in key agreement, the temporary secret key r_i should not be involved in the computation before the exchange of information. So the random value r_i can be reconfirmed in the online phase. This idea motivates us to propose a one-to-many key agreement scheme for SM9, which is used to achieve the fast computation of the SM9-KA algorithm before the exchange of information between the two parties.

We consider determining a part of the temporary secret key, defined as s, out of the heavy computations in the offline phase. Then we confirm another part of the temporary secret key, r_i, in the online phase. As long as it is guaranteed that each time the r_i determined in the present phase is random, the value of the secret that we want to agree, sr_i, each time will be random as well. Based on this idea, we separate the similar ID header part in SM9-KA into three parts:

$$\begin{cases} C_1 & = [s]([w]P_1 + [t]P_{pub}) \\ C_2 & = s(r_i ID_i - w) \bmod p \\ C_3 & = s(r_i - t) \bmod p \end{cases}$$

The party who receives the three parts can set the ID header by this way:

$$\begin{aligned} R_i &= C_1 + [C_2]P_1 + [C_3]P_{pub} \\ &= [s]([w]P_1 + [t]P_{pub}) + [s(r_i ID_i - w)]P_1 + [s(r_i - t)]P_{pub} \\ &= [sr_i]([ID_i]P_1 + P_{pub}) \end{aligned}$$

Therefore, the elliptic curve value, which would have been required to start the calculation at the information exchange stage, was split into two parts:

$$\begin{cases} C_{off} & = C_1 \\ C_{on} & =< C_2, C_3 > \end{cases}$$

Values computed in the offline phase can be sent in advance and used multiple times, and values in the online phase have very little overhead, allowing for timely response. This greatly reduces the computational burden at the time of interaction.

4.2 A strategy of Non-Reciprocal Improvement

In this section we consider further efficiency optimization approach for SM9-KA.

In the SM9-KA scheme, the most time-consuming computation is the operation on bilinear pairings. Although some of the pairing operations can be performed in advance to improve efficiency, at least two pairing operations cannot be avoided in the whole scheme. Considering that the computational resources of the two parties conducting the key agreement may be unbalanced, we design a way to allow one party to hand over its own pairing computation to the other party to do it, forming a kind of non-reciprocal key agreement protocol.

Supposing that a party U_i in the protocol should receive another party U_j's message R_j and compute $g_t = e(R_j, D_i), t \in \{1, 2\}$. This pairing operation cannot avoid because of the authentication. But if U_i does not have the ability to compute pairing operations, it can use the following steps to leave the pairing operation to the other side.

(1) U_i randomly chooses $x \in \mathbb{Z}_p^*$, and computes $\sigma_i = [x]D_i$. Then he sends σ_i to U_j.
(2) U_j computes $R_j = [H_1(ID_i)]P_1 + P_{pub}$ as usual. Then he uses σ_i to compute and send $T_j = e(R_j, \sigma_i)$.
(3) When U_i receives T_j, he can use x stored locally to compute required value $g_t = T_j^{1/x}, t \in \{1, 2\}$.

In this way, the pairing operation that U_i would otherwise have to perform can be given to U_j. However, U_j itself must perform the pairing operation that cannot be avoided, so it must perform two pairing operations. Although the operations of both sides are unequal in this way, it has certain usage scenarios, such as user U_i being a lightweight mobile device that does not have the ability to perform complex operations, while user U_j is a server.

4.3 Non-Reciprocal Fast Key Agreement Scheme Based on SM9

In this subsection, we combine the ideas of the above two strategies, thus forming a non-reciprocal fast key agreement scheme based on SM9. The specific key agreement protocol is as follows:

Setup(1^λ): The PKG generates the system parameters as follows when inputting a secure parameter λ:

(1) Choose a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p. Randomly choose generators $P_1 \in \mathbb{G}_1, P_2 \in \mathbb{G}_2$.
(2) Randomly choose $s \in \mathbb{Z}_p^*$ and compute the master public key $P_{pub} = [s]P_1$.
(3) Calculate $v = e(P_{pub}, P_2)$ in advance.
(4) Choose a hash function H_1 and an additional bit hid.
(5) Output public parameters:$Params =< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, P_{pub}, v, H_1, hid >$. So the master private key is $msk = s$.

PrivateKeyExtract$(Params, msk, ID_i)$**:** When PKG receives a party U_i's identify ID_i, it uses the public parameters and the master secret key msk to generate the user's private key:

$$D_i = [\frac{s}{s + H_1(ID_i||hid, p)}]P_2$$

Note that if $s + H_1(ID_i||hid, p) \equiv 0 \bmod r$, the PKG should react the **Setup** step and change a new master secret key. The PKG sends the key to the user over a secure channel after it generates the user's private key.

OfflineTransfer: The client should first generate values offline and send them to the servers from a safe channel. So the client should act as follows:

(1) Randomly choose $s, w, t \in \mathbb{Z}_p^*$ and compute $C_{off-C} = [s]([w]P_1 + [t]P_{pub})$.
(2) Randomly choose $x \in \mathbb{Z}_p^*$ and compute $\sigma_C = [x]D_C$.
(3) Send $< C_{off-C}, \sigma_C >$ to the servers.

Upon one server receives the values, it will randomly choose a number $r_{S_C} \in \mathbb{Z}_p^*$ and calculate a value of the client offline in advance to prepare for message exchange:

$$T_{S_C} = e([r_{S_C}]([H_1(ID_C||hid, p)]P_1 + P_{pub}), \sigma_C)$$

Now it can store the values: $< ID_C, T_{S_C}, r_{S_C}, C_{off-C} >$. Note that if a server wants to change the value T_{S_C}, it should not recalculate it from the same way. It can just choose a number $r'_{S_C} \in \mathbb{Z}_p^*$ and calculate $T'_{S_C} = T_{S_C}^{r'_{S_C}/r_{S_C}}$ and restore the new T'_{S_C}. This action is more faster than original.

MessageExchange: (Optional value is neglected.) When the client U_C wants to communicate with the server U_S, U_C randomly selects $r_C \in \mathbb{Z}_p^*$ and uses s_C, w_C, t_C he stored and compute:

$$\begin{cases} C_{C-1} & = s_C(r_C H_1(ID_S||hid, p) - w_C) \bmod p \\ C_{C-2} & = s_C(r_C - t_C) \bmod p \end{cases}$$

Then he sends the two values to U_S.

When U_S receives the messages U_C sends, it sends $T_S = T_{S_C}$ to the client.

SessionKeyGeneration: For the client U_C, he should compute the session key as follows:

$$\begin{cases} g_1 = T_S^{1/x} \\ g_2 = v^{s_C r_C} \\ g_3 = g_2^{s_C r_C} \end{cases}$$

$$SK_A = KDF(ID_C||ID_S||C_{off-C}||C_{C-1}||C_{C-2}||T_S||g_1||g_2||g_3, l)$$

While for the server U_S, he should compute the session key as follows:

$$\begin{cases} g_1' = v^{rs_C} \\ g_2' = e(C_{off-C} + [C_{C-1}]P_1 + [C_{C_2}]P_{pub}, D_S) \\ g_3' = g_2''^{rs_C} \end{cases}$$

$$SK_B = KDF(ID_C||ID_S||C_{off-C}||C_{C-1}||C_{C-2}||T_S||g_1'||g_2'||g_3', l)$$

5 SECURITY ANALYSIS

5.1 Proof of Scheme Security

Theorem 1. *Supposing that H_1 and KDF are random oracles, if there exists a PPT adversary who can break our protocol with non-negligible advantage $\epsilon(k)$ in time $t(k)$, there exists a algorithm C solving $Gap-\tau-BCAA1$ problem with advantage $Adv_C(k) \geq \frac{\epsilon(k)}{(q_1+1)\cdot q_o}$ within a running time $t_C \leq t(k) + O(q_2 \cdot q_o \cdot O)$. The adversary has the ability to query H_1 q_1 times and KDF q_2 times, and it can create q_o oracles.*

Proof. Now we will prove that if there exists an adversary who can break our scheme, we can create a non-negligible advantage algorithm C to solve $Gap-\tau-BCAA1$ problem in probabilistic polynomial time (PPT).

Given an instance of the $Gap-\tau-BCAA1$ problem $(P_1, P_2, [x]P_1, h_0, (h_1, [\frac{x}{h_1+x}]P_2), ..., (h_{q_1}, [\frac{x}{h_{q_1}+x}]P_2))$, C creates system parameters from it: $Params =< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, [x]P_1, e([x]P_1, P_2), hid >$. Note that the master secret key x is unknown to C.

The functions H_1 and KDF are constructed from hash functions. They are simulated as random oracles controlled by C. C chooses $1 \leq I \leq q_1 + 1$ and $1 \leq J \leq q_0$, and interacts with the adversary $\mathcal{A}$ as follows:

First, C chooses a party U_S as the sender and maintains a list $Offline^{list}$: $(t, s^t, w^t, t^t, x^t, C^t_{off-S}, \sigma^t_S)$. If $t = J$, C lets $s^t = 1, w^t = 0$ and randomly chooses $t^t \in \mathbb{Z}^*_p$. Or otherwise, C randomly selects $s^t, w^t, t^t \in \mathbb{Z}^*_p$. Then it computes $C_{off-S} = [s^t]([w^t]P_1 + [t^t]P_{pub})$. After that, C randomly chooses $x^t \in \mathbb{Z}^*_p$ and computes $\sigma^t_S = [x^t]D_S$. Finally, C gives all $(t, C^t_{off-S}, \sigma^t_S)$ to the adversary $\mathcal{A}$ and saves $(t, s^t, w^t, t^t, x^t, C^t_{off-S}, \sigma^t_S)$ into the list $Offline^{list}$.

While finishing the actions above, C can answer the adversary $\mathcal{A}$'s queries:

- $H_1(ID_i)$ **Query:** C maintains a list H_1^{list} with entries (ID_i, h_i, D_i). When the adversary queries about the ID_i's H_1, C responds as follows:
  - If (ID_i, h_i, D_i) can be found with key ID_i in H_1^{list}, C responds $H_1(ID_i) = h_i$.
  - Else if $i = I$, C stores $(ID_I, h_0, \perp)$ in H_1^{list} and responds $H_1(ID_I) = h_0$.
  - Otherwise, C randomly chooses a $0 < i \leq q_1$ from the instance of $Gap-\tau-BCAA1$ which has not yet been selected and inserts $(ID_i, h_i, [\frac{x}{h_i+x}]P_2)$ into H_1^{list}. Then it responds $H_1(ID_i) = h_i$.
- $KDF(ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3)$ **Query:** C maintains a list of $(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3 >, \zeta_t)$ named KDF^{list} and responds as follows:
  - If an entry $< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3 >$ can be found in KDF^{list}, C responds ζ_t.
  - Otherwise, C searches entries $(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j >, r^t, \zeta^t)$ whose key is $< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j >$ from the list Λ and acts as follows:
    - Let $R = C_{off-i} + [C_{i-1}]P_1 + [C_{i-2}]P_{pub}$ and $T = g_1^t$, query O_{DBIDH} about $([x]P_1, P_2, [h_0 + x]P_1, R, T)$.
    - When O_{DBIDH} returns 1, C judges if $g_3^t = T^{r^t}$. If it equals, C removes this entry from Λ and inserts $(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3 >, \zeta^t)$ into KDF^{list}. It responds ζ^t.
  - Otherwise, if there is no entry found above or no entry satisfied the condition, it means that this is a new entry. C randomly chooses $\zeta_t \in \{0,1\}^l$ and inserts $(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3 >, \zeta_t)$ into KDF^{list} and responds ζ_t.
- $Corrupt(ID_i)$ **Query:** C goes through H_1^{list} to find ID_i. If there not exists ID_i, C queries $H_1(ID_i)$. Else C checkes the value D_i: If $D_i \neq \perp$, C responds D_i; otherwise C aborts the game. (**Event 1**)
- $Send(\prod^t_{i,j}, R)$ **Query:** C maintains a list $(\prod^t_{i,j}, r^t_{i,j}, tran^t_{i,j})$ and responds as follows:
  - If $i \neq S$ or $j \neq S$, C responds that the session is rejected. (Note that the sender in our protocol can only be S.)
  - Else if $t \neq J$, C responds with the oracle's identity:
    - If $R = \lambda$, the oracle is the sender's. C finds the t's values (t, s^t, w^t, t^t) from $Offline^{list}$ and randomly selects randomly $r^t_{i,j} \in \mathbb{Z}^*_p$. Then it computes and responds $tran^t_{i,j} = (C^t_{i,j-1}, C^t_{i,j-2}) = (s^t(r^t_{i,j}H_1(ID_j) - w^t), s^t(r^t_{i,j} - t^t))(\mathrm{mod}p)$.
    - Otherwise, the oracle belongs to the respond party. C finds the t's values σ^t_S from $Offline^{list}$ adn randomly chooses $r^t_{i,j} \in \mathbb{Z}^*_p$. After that it computes and responds $trans^t_{i,j} = T_i = e([r^t]([H_1(ID_i)]P_1 + P_{pub}), \sigma^t_i)$.
  - Otherwise, C queries $H_1(ID_j)$ and responds as follows:
    - If $D_j \neq \perp$, C aborts. (**Event 2**)
    - Otherwise, after C goes through the $Offline^{list}$ list and gets t^t, it randomly choose $y \in \mathbb{Z}^*_p$ and responds $tran^t_{i,j} = (C^t_{i,j-1}, C^t_{i,j-2}) = (y, -t^t)(\mathrm{mod}p)$
- $Reveal(\prod^t_{i,j})$ **Query:** C maintains a list Λ: $(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j >, r^t, \zeta^t)$, and responds as follows:
  - If $t = J$ or the oracle matches $\prod^J_{i,j}$ wants to be revealed, C aborts. (**Event 3**)
  - C goes through H_1^{list} and finds ID_i to get D_i.
  - If the session to be revealed is the sender's, C computes $g_1 = e([x]P_1, P_2)^{s_i r^t_i}, g_2 = T_j^{1/x_i}, g_3 = g_2^{s_i r^t_i}$ and responds $KDF(ID_i, ID_j, C_{off-i}, C_{i-1}, C_{i-2}, T_j, g_1, g_2, g_3)$.
  - Otherwise, if $D_i \neq \perp$, C computes $e(C_{off-S} + [C_{j-1}]P_1 + [C_{j-2}]P_{pub}, D_i), g_2 = e([x]P_1, P_2)^{r^t_{i,j}}, g_3 = g_1^{r^t_{i,j}}$ and responds $KDF(ID_j, ID_i, C_{off-j}, C_{j-1}, C_{j-2}, T_i, g_1, g_2, g_3)$.
  - Otherwise, because C cannot compute anything about D_I, it should goes through KDF^{list} and finds $< ID_j, ID_i, < C_{off-j}, C_{j-1}, C_{j-2} >, T_i, e([x]P_1, P_2)^{r^t}, *, * >$. Note that

∗ means matching any value. After C gets each $(g_1^t, g_2^t, g_3^t, \zeta_t)$, it checks as follows:
  - ∗ Let $R = C_{off-j} + [C_{j-1}]P_1 + [C_{j-2}]P_{pub}, T = g_2^t$ and queries O_{DBIDH} with $([x]P_1, P_2, [h_0 + x]P_1, R, T)$.
  - ∗ If O_{DBIDH} returns 1 and $g_3^t = T^{r^t}$, C responds ζ_t.
- Otherwise, C randomly chooses $\zeta^t \in \{0,1\}^l$ and inserts$(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j >, r^t, \zeta^t)$ into Λ. After that, C responds ζ^t.

- $Test(\prod_{i,j}^t)$ **Query:** If $t \neq J$ or the oracle matches $\prod_{i,j}^t$ has been revealed, C aborts the game. (**Event 4**) Otherwise, C randomly chooses $\zeta \in \{0,1\}^l$ and responds.

After $\mathcal{A}$ finishes all the queries and responds its guess, C goes through the KDF^{list} and finds all the entries $(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3 >, \zeta_t)$ and compares them with the oracles in *Reveal*. If the oracle is a sender, let $R = C_{off-i} + [C_{i-1}]P_1 + [C_{i-2}]P_{pub}, T = g_1^t$. Otherwise let $R = C_{off-j} + [C_{j-1}]P_1 + [C_{j-2}]P_{pub}, T = g_2^t$. Then C queries O_{DBIDH} about $([x]P_1, P_2, [h_0 + x]P_1, R, T)$:

- If O_{DBIDH} returns 1, C responds $T^{1/y}$ as the answer of the $Gap - \tau - BCAA1$ problem.
- Otherwise if no entry is found, C lost. **(Event 5)**

Claim. $Pr[\overline{Event5}] \geq \epsilon(k)$.

This proof is the same as Claim 2 in [5]. We skip the detail.

Let **Event 6** be that adversary $\mathcal{A}$ indeed chooses oracle $\prod_{i,j}^J$ as the challenger oracle in the attack. That is to say, Event 1, 2, 3, 4 would not happen. In that, $Pr[\overline{Event2}] = \frac{1}{(q_1+1)}, Pr[\overline{Event4}] = \frac{1}{q_o}$. So,

$$\begin{aligned} Pr[Event6] &= Pr[\overline{Event1 \vee Event2 \vee Event3 \vee Event4}] \\ &= \frac{1}{(q_1+1)\cdot q_o} \end{aligned}$$

So,

$$Pr[\mathcal{A}wins] = Pr[Event6 \wedge \overline{Event5}] \geq \frac{\epsilon(k)}{(q_1+1)\cdot q_o}$$

5.2 Proof of Forward Security

Theorem 3. *Supposing that KDF is a random oracle, if there exists a PPT adversary who can break our protocol with non-negligible advantage $\epsilon(k)$ in time $t(k)$, there exists a algorithm C solving $\psi - BDH$ problem with advantage $Adv_C(k) \geq \frac{\epsilon(k)}{q_1 \cdot q_o}$ within a running time $t_C \leq t(k)$. The adversary has the ability to query KDF q_1 times, and it can create q_o oracles.*

Proof. Now we will prove that if there exists a PPT adversary who can break our scheme, we can create a non-negligible advantage algorithm C to solve $\psi - BDH$ problem.

Given an instance of the $\psi - BDH$ problem $(P_1, P_2, [\alpha]P_1, [\beta]P_1)$, and there does not exist a map that $P_i = \phi(P_j)$, for $i, j \in \{1,2\}$ and $i \neq j$, C creates system parameters from it: $Params =< \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, [x]P_1, e([x]P_1, P_2), hid >$. Note that the master secret key x is known to C.

In its operational form, function KDF comprises a hash function and is implemented in a manner consistent with the definition of a random oracle under the control of C. C selects an integer $1 \leq J \leq q_0$, and engages in a dialogue with the adversary $\mathcal{A}$ in accordance with the following protocol:

First, C chooses a party U_S as the sender and maintains a list $Offline^{list}$: $(t, s^t, w^t, t^t, x^t, C_{off-S}^t, \sigma_S^t)$. C randomly chooses $x^t \in \mathbb{Z}_p^*$, and computes $\sigma_S^t = [x^t][\frac{x}{H_1 ID_S + x}]P_2$. If $t \neq J$, C randomly selects $s^t, w^t, t^t \in \mathbb{Z}_p^*$, and computes $C_{off-S} = [s^t]([w^t]P_1 + [t^t]P_{pub})$. Otherwise, C randomly chooses $C_{S-1}, C_{S-2} \in \mathbb{Z}_p^*$ and computes $C_{off-S}^t = [\alpha]P_1 - [C_{S-1}]P_1 - [C_{S-2}]P_{pub}$ and saves $(t, \perp, C_{S-1}, C_{S-2}, x^t, C_{off-S}^t, \sigma_S^t)$ into $Offline^{list}$. After that, C gives all $(t, C_{off-S}^t, \sigma_S^t)$ to the adversary $\mathcal{A}$ and saves $(t, s^t, w^t, t^t, x^t, C_{off-S}^t, \sigma_S^t)$ except $t = J$ into the list $Offline^{list}$.

While finishing the actions above, C can answer the adversary $\mathcal{A}$'s queries:

- $KDF(ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3)$ **Query:** C maintains a list KDF^{list} whitch has entries like $(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3 >, \zeta_t)$, and answers as follows:
  - If an entry $< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3 >$ can be found in KDF^{list}, C responds ζ_t.
  - Otherwise, C randomly chooses $\zeta_t \in \{0,1\}^l$ and inserts $(< ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, T_j, g_1, g_2, g_3 >, \zeta_t)$ into KDF^{list}. Then it responds ζ_t.
- $Corrupt(ID_i)$ **Query:** C computes $D_i = [\frac{x}{H_1(ID_i)+x}]P_2$ and returns D_i.
- $Send(\prod_{i,j}^t, R)$ **Query:** C maintains a list $(\prod_{i,j}^t, r_{i,j}^t, tran_{i,j}^t)$ and responds as follows:
  - If $i \neq S$ or $j \neq S$, C responds that the session is rejected. (Note that the sender in our protocol can only be S.)
  - Else if $t \neq J$, C responds with the oracle's identity:
    - ∗ If $R = \lambda$, the oracle is the sender's. C finds the t's values (t, s^t, w^t, t^t) from $Offline^{list}$ and randomly selects randomly $r_{i,j}^t \in \mathbb{Z}_p^*$. Then it computes and responds $tran_{i,j}^t = (C_{i,j-1}^t, C_{i,j-2}^t) = (s^t(r_{i,j}^t H_1(ID_j) - w^t), s^t(r_{i,j}^t - t^t))(\mathrm{mod} r)$.
    - ∗ Otherwise, the oracle belongs to the respond party. C finds the t's values σ_S^t from $Offline^{list}$ adn randomly chooses $r_{i,j}^t \in \mathbb{Z}_p^*$. After that it computes and responds $trans_{i,j}^t = T_i = e([r^t]([H_1(ID_i)]P_1 + P_{pub}), \sigma_i^t)$.
  - Otherwise, it means that $t = J$.
    - ∗ If $R = \lambda$, C goes through $Offline^{list}$ and finds $(J, \perp, w^t, t^t)$. After that, it lets $C_{S-1} = w^t, C_{S-2} = t^t$ and responds $tran_{i,j}^J = (C_{S-1}, C_{S-2})$.
    - ∗ Otherwise, C goes through $Offline^{list}$ and finds $(J, C_{off-S}^J, \sigma_S^J)$. After that, it tests if $R = (C_{j-1}^t, C_{j-2}^t)$ can satisfy $C_{off-S}^J + [C_{j-1}^t]P_1 + [C_{j-2}^t]P_{pub} = [\alpha]P_1$. If not, C aborts the game. (**Event 1**) Otherwise, C computes and responds $tran_{i,j}^J = T_i = e([\beta]P_1, \sigma_j^J)$.
- $Reveal(\prod_{i,j}^t)$ **Query:** C responds as follows:
  - If $t = J$, C aborts the game. (**Event 2**)
  - Otherwise, C finds the t's Offline values $(t, s^t, w^t, t^t, x^t, C_{off-S}^t, \sigma_S^t)$ from $Offline^{list}$ and responds as follows:

Table 1: Theoretical Compairation

Step	Traditional SM9	SM9-OO-KA	Ours
Offline(Client)	/	$n \cdot (2E + m_c)$	$2E + 2m_c + E$
Offline(Server)	/	$n \cdot (2E + m_c)$	$1P + 2E + 1m_c$
MessageExchange(Client)	$n \cdot 2E$	$n \cdot 2m_c$	$n \cdot 3m_c$
MessageExchange(Server)	$n \cdot 2E$	$n \cdot 2m_c$	$(n-1) \cdot m_e$
LocalKeyGeneration(Client)	$n \cdot (1P + 2m_e)$	$n \cdot (1P + 1E + 2m_e)$	$n \cdot 3m_e$
LocalKeyGeneration(Server)	$n \cdot (1P + 2m_e)$	$n \cdot (1P + 1E + 2m_e)$	$n \cdot (1P + 2E + 2m_e)$

  - If the oracle is from the sender, C computes $g_1 = e([x]P_1, P_2)^{s^t r^t_{i,j}}, g_2 = T_j^{1/x^t}, g_3 = g_2^{s^t r^t_{i,j}}$ and responds $KDF(ID_i, ID_j, C_{off-S}, C_{i-1}, C_{i-2}, T_j, g_1, g_2, g_3)$.
  - Otherwise, C computes $g_1 = e(C_{off-S} + [C_{j-1}]P_1 + [C_{j-2}]P_{pub}, [\frac{s}{H_1(ID_i)+s}]P_2), g_2 = e([x]P_1, P_2)^{r^t_{i,j}}, g_3 = g_1^{r^t_{i,j}}$ and responds $KDF(ID_j, ID_i, C_{off-j}, C_{j-1}, C_{j-2}, T_i, g_1, g_2, g_3)$.
- $Test(\prod^t_{i,j})$ **Query:** If $t \neq J$ or the oracle matches $\prod^t_{i,j}$ has been revealed, C aborts the game. (**Event 3**) Otherwise, C randomly chooses $\zeta \in \{0,1\}^l$ and responds.

After $\mathcal{A}$ finishes all the queries and responds its guess, C goes through the KDF^{list} and finds all the entries satisfied $(ID_i, ID_j, < C_{off-i}, C_{i-1}, C_{i-2} >, *, e([\beta]P_1, [\frac{x}{H_1(ID_i)+x}]P_2), e([\alpha]P_1, [\frac{x}{H_1(ID_j)+x}]P_2), *)$ and $C_{off-i} + [C_{i-1}]P_1 + [C_{i-2}]P_{pub} = [\alpha]P_1$. Then it randomly chooses one's g_3, and computes $X = g_3^{(H_1(ID_i)+x)(H_1(ID_j)+x)}$ as the answer of $\psi - BDH$ problem. If the adversary $\mathcal{A}$ queries values about the J's oracle, C can get the correct answer to the $\psi - BDH$ problem with probablity at least $1/q_1$.

Let **Event 4** be that adversary $\mathcal{A}$ indeed chooses oracle $\prod^J_{i,j}$ as the challenger oracle in the attack. That is to say, Event 1, 2, 3 would not happen. In that, $Pr[\overline{Event3}] \geq \frac{1}{q_o}$, so:

$$Pr[Event4] = Pr[\overline{Event1 \vee Event2 \vee Event3}] \geq \frac{1}{q_o}$$

So:

$$Pr[\mathcal{A}wins] \geq Pr[Event4] \cdot \frac{\epsilon(k)}{q_1} \geq \frac{\epsilon(k)}{q_o \cdot q_1}$$

All the proofs are completed.

6 COMPARISON

In this section we will illustrate the SM9-OO-KA by comparing it with the original SM9-KA, and we will illustrate the advantages of our scheme with theoretical and practical data. Let E be a scalar multiplication operation on the group of elliptic curves $\mathbb{G}$, m_c be a multiplication operation on the group of integers $\mathbb{Z}^*_p$, P be a pairing mapping operation, and m_e be a power operation. Since the pointwise addition on elliptic curves and the addition on the group of integers have fewer operations than the previous two, they are ignored here.

The results of the theoretical analysis are shown in the Table 1. Supposing that there are n interactions.

In Table 1 we can see that although the SM9-OO-KA can perform complex operations offline, it should still compute n times. The values of the offline computation should be stored locally ready for online communication. After the message exchange phase, each part using the SM9-OO-KA should compute one more point-wise multiplication operation than those using the SM9-KA. Overall, although it enables timely response capability, the SM9-OO-KA is not a major upgrade over the original SM9-KA.

However, in our scheme we should only compute once in the Offline phase. The rapid computational capacity of the client's online phase in SM9-OO-KA is maintained in our proposed scheme, and the client's local key generation expenditure subsequent to the key exchange phase is reduced. In addition, the server in our scheme trades space for time by computing offline, which is a small increase in total computation, although it has to take on more computation.

An experimental analysis is also given below. Table 2 is the experimental environment and Figure 1, Figure 2 are the results. Supposing that there are 100 interactions.

Table 2: Experimental Environment

CPU	OS	RAM	Compiler & Library
Intel(R) i5-10400	Ubuntu 22.04.3	16G DDR4	GCC & PBC

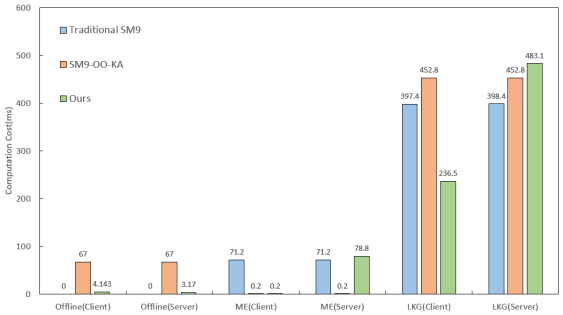

Figure 1: Partical Experimental Results

We can see in Figure 1 that our scheme can achieve a significant reduction in the computation time in the offline phase of SM9-OO-KA, while maintaining the fast computational capability of the client's online phase, providing a better solution to the problem of timely response for high-speed devices. In addition, the local key

generation phase of the transmitter is half the cost of SM9-KA and SM9-OO-KA. The server's local key generation cost increases only slightly.

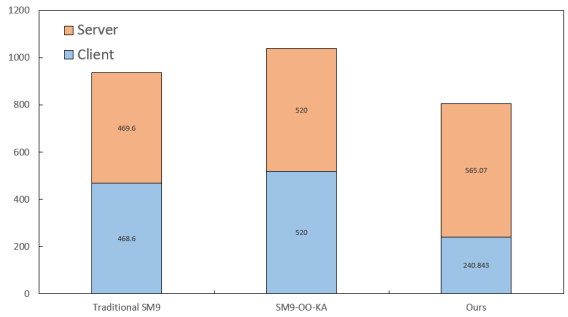

Figure 2: Total Experimental Results

Since all the required values are already obtained in the Offline phase, it is perfectly possible for the server to consume offline pre-calculations and store them after receiving the Offline values, as in the case of the SM9-OO-KA. As shown in the figure.

From Figure 2 we can see that our scheme can almost halve the computation on the client side, while the increase in server-side computational overhead is limited, making it more suitable for lightweight devices. This is the advantage of the non-reciprocity of our scheme: in the case where the reduction of the total elapsed time implies a replacement of the scheme, the transformations proposed by us can allow the scheme to be better adapted to the client-server model.

In summary, the comparison shows that our scheme successfully achieves the purpose of two real-world scenarios: timely response and solutions for resource-constrained devices.

7 CONCLUSION

In this paper, an SM9-based non-reciprocal fast key agreement algorithm is proposed by combining online/offline algorithms, which effectively solves the problem of computational volume of SM9-KA before sending messages and the problem of pairing operations, and greatly improves the efficiency of performing key agreement. The efficacy of this scheme is demonstrated in the paper through a security proof and an efficiency comparison with existing schemes. This illustrates the feasibility of this scheme. This scheme has a good application prospect in the lightweight client-to-server key agreement scenario.

REFERENCES

[1] Abigail Akosua Addobea, Jun Hou, Qianmu Li, and Huaizhi Li. 2020. MHCOOS: An Offline-Online Certificateless Signature Scheme for M-Health Devices. *Sec. and Commun. Netw.* 2020 (jan 2020), 12 pages.

[2] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. 1997. Key agreement protocols and their security analysis. In *Crytography and Coding*, Michael Darnell (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 30–45.

[3] Liqun Chen and Zhaohui Cheng. 2005. Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme. In *Cryptography and Coding*, Nigel P. Smart (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 442–459.

[4] Zhaohui Cheng. 2017. The SM9 Cryptographic Schemes. Cryptology ePrint Archive, Paper 2017/117.

[5] Zhaohui Cheng. 2019. Security Analysis of SM9 Key Agreement and Encryption. In *Information Security and Cryptology*, Fuchun Guo, Xinyi Huang, and Moti Yung (Eds.). Springer International Publishing, Cham, 3–25.

[6] Sherman S. M. Chow, Joseph K. Liu, and Jianying Zhou. 2011. Identity-based online/offline key encapsulation and encryption. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (Hong Kong, China) *(ASIACCS '11)*. Association for Computing Machinery, New York, NY, USA, 52–60.

[7] Jie Cui, Han Zhou, Yan Xu, and Hong Zhong. 2019. OOABKS: Online/offline attribute-based encryption for keyword search in mobile cloud. *Information Sciences* 489 (2019), 63–77.

[8] Shimon Even, Oded Goldreich, and Silvio Micali. 1990. On-Line/Off-Line Digital Signatures. In *Advances in Cryptology — CRYPTO' 89 Proceedings*, Gilles Brassard (Ed.). Springer New York, New York, NY, 263–275.

[9] Fuchun Guo and Yi Mu. 2008. Optimal Online/Offline Signature: How to Sign a Message without Online Computation. In *Provable Security*, Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 98–111.

[10] Fuchun Guo, Yi Mu, and Zhide Chen. 2008. Identity-Based Online/Offline Encryption. In *Financial Cryptography and Data Security*, Gene Tsudik (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 247–261.

[11] Fuchun Guo, Yi Mu, and Willy Susilo. 2011. Efficient Online/Offline Signatures with Computational Leakage Resilience in Online Phase. In *Information Security and Cryptology*, Xuejia Lai, Moti Yung, and Dongdai Lin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 455–470.

[12] Rui Guo. 2022. Design of Lightweight UAV Network Authentication Key Negotiation Protocol Based on State Secret Algorithm (in chinese).

[13] Jayaprakash Kar. 2012. Provably Secure Online/Off-line Identity-Based Signature Scheme forWireless Sensor Network. Cryptology ePrint Archive, Paper 2012/162.

[14] Jianchang Lai, Xinyi Huang, Debiao He, and Wei Wu. 2021. Provably Secure Online/Offline Identity-Based Signature Scheme Based on SM9. *Comput. J.* 65, 7 (03 2021), 1692–1701.

[15] Jianchang Lai, Yi Mu, Fuchun Guo, and Willy Susilo. 2015. Improved Identity-Based Online/Offline Encryption. In *Information Security and Privacy*, Ernest Foo and Douglas Stebila (Eds.). Springer International Publishing, Cham, 160–173.

[16] Joseph K Liu, Joonsang Baek, Jianying Zhou, Yanjiang Yang, and Jun Wen Wong. 2010. Efficient online/offline identity-based signature for wireless sensor network. *International Journal of Information Security* 9 (2010), 287–296.

[17] Joseph K. Liu and Jianying Zhou. 2009. An Efficient Identity-Based Online/Offline Encryption Scheme. In *Applied Cryptography and Network Security*, Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 156–167.

[18] Sharmila Deva Selvi S, Sree Vivek S, and Pandu Rangan C. 2010. Identity Based Online/Offline Encryption Scheme. Cryptology ePrint Archive, Paper 2010/178.

[19] S. Sharmila Deva Selvi, S. Sree Vivek, and C. Pandu Rangan. 2011. Identity Based Online/Offline Encryption and Signcryption Schemes Revisited. In *Security Aspects in Information Technology*, Marc Joye, Debdeep Mukhopadhyay, and Michael Tunstall (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 111–127.

[20] Fei TANG, Guowei LIN, and Jinyong SHAN. 2022. Additive homomorphic encryption scheme based on state secret SM2 and SM9 (in chinese). *Journal of Cryptologic Research* 9, 03 (2022), 535–549.

[21] Andrew Chi-Chih Yao and Yunlei Zhao. 2013. Online/Offline Signatures for Low-Power Devices. *IEEE Transactions on Information Forensics and Security* 8, 2 (2013), 283–294.

[22] Mingmei Zheng, Shao-Jun Yang, Wei Wu, Jun Shao, and Xinyi Huang. 2018. A New Design of Online/Offline Signatures Based on Lattice. In *Information Security Practice and Experience*, Chunhua Su and Hiroaki Kikuchi (Eds.). Springer International Publishing, Cham, 198–212.