Common Knowledge Based Secure Generation and Exchange of Symmetric Keys

Hexiong Chen[1], Jiaping Wu[2], Wei Guo[1], Feilu Hang[1], Zhenyu Luo[1], and Yilin Wang[2]
(Corresponding author: Jiaping Wu)

Information Center of Yunnan Power Grid Co. Ltd[1]
Kunming, Yunnan 650000, China
Yangtze Delta Region Institute (Quzhou), University of Electronic Science and Technology of China[2]
Quzhou, Zhejiang 324000, China
Email: 13981671425@163.com

(Received Apr. 27, 2022; Revised and Accepted Dec. 3, 2022; First Online Dec. 13, 2022)

Abstract

Man-in-the-middle attacks bring severe security threats to the Diffie-Hellman (DH) based symmetric key generation and exchange. This paper proposed a common knowledge-based technique for secure key generation and exchange to ensure the security of the key generation between the communication peers. A message matching mechanism with low cost is designed without interfering with or changing the standard communication mechanism. Therefore, the common knowledge about the communication peers' historical communication messages is obtained using the shared knowledge generation and verification algorithm. Based on the common knowledge and the proposed sizeable prime number generation algorithm, DH symmetric key agreement with low information exchange is implemented to improve the security in generating the symmetric key. Theoretical analysis and simulation results show that the proposed schemes can considerably improve the security in the process of the symmetric key generation with low resource cost.

Keywords: Common Knowledge; Consistency Verification; Key Exchange; Message Matching Scheme; Symmetric Key

1 Introduction

In recent years, with the booming development of information technology, the scale of information networks has continued to expand, and the applications of information networks have become increasingly abundant. These applications generally have the characteristics of a large size of transmitted data and strong real-time communication requirements. The use of a symmetric encryption mechanism to encrypt data can better meet application requirements.And the key generation and exchange are crucial technologies of symmetric encryption. In 1976, Diffie and Hellman jointly proposed the Diffie-Hellman key exchange(DHKE) protocol to securely generate symmetric keys for the the communication peers [11], which has been widely used in network security protocols such as SSL (Security Socket Layer) and IPsec (Internet Protocol Security). However, the DHKE protocol has always had the problem of MITM attacks in practice [2], which leads to the disclosure of session content and cannot guarantee the security of the communication process.

To solve the problem of MITM attack of DHKE protocol, many studies have been done in academia and industry. In 2010, Yoon *et al.* [12] proposed a secure DHKE protocol based on Chebyshev polynomials and chaotic mapping. Since then, some researchers have extended the application of this protocol to achieve identity authentication in different scenarios [1,8,17,19]. But these studies increased interactions and communication cost. In 2014, Shen *et al.* [24] proposed a technique for communication key transmission between device-to-device (D2D) based on DHKE protocol. In 2015, Khader *et al.* [16] analyzed the MITM attack problem of the DHKE protocol and proposed a scheme using the Geffe binary number generator, to improve the key generation mechanism and resist the MITM attacks. In 2019, Zhang *et al.* [26] proposed that both parties used shared secret matrix eigenvalues for key agreement. In 2021, Shen *et al.* [23] presented a novel in-band solution for defending the MITM attack during the key establishment process for wireless devices. Though these studies have enhanced the security of the DHKE mechanism, they have led to high communication costs or long key generation time.

The research of protecting key exchange information is as follows. In [7], Bui et.al proposed a key exchange protocol using blockchains and other public ledger structures. In [21], Naher *et al.* proposed a DHKE protocol based on a shared CRC (Cyclic Redundancy Check) polynomial, which can detect MITM attacks in the process of key information exchange, but it cannot prevent MITM attacks. In [25], Thwe *et al.* proposed to protect the key exchange information of the DHKE protocol by hash function, to resist the MITM attacks encountered during the key generation process. In 2020, Chunka [9] improved the mechanism of the DHKE protocol and prevented the key exchange information from being tampered with by using digital signature technology. Ali *et al.* [4] proposed an approach to defend against attacks by generating a hash of each value transmitted over the network. However, these studies have high computational complexity in protecting key exchange information.

Meanwhile, the key generation mechanism based on communication context has emerged. In 2020, Dar *et al.* [10] proposed to calculate the level of confidentiality of each message based on context-aware computing, and select the optimal encryption algorithm according to the confidentiality level. This study has reduced the resource cost and time delay of the encryption and decryption process. However, context-aware computing has greater computational complexity. In [15, 20], the authors proposed to perceive communication data using context-aware computing in real time and combine it with attribute encryption technology to achieve access control encryption for communication data. However, these studies only use the key attributes in the message for contextual computing, such as time, location, and device status. This study is vulnerable to attacks third-party. Consequently, if the computational complexity of the context-awareness computing is great, it will increase the encryption and decryption time. When the message is altered, diverted, or leaked, it will bring threats to the security of future communication by using contextual computing in historical messages to encrypt new messages.

In order to reduce the risk of key leakage and improve the security of communication, this paper proposes a common knowledge-based symmetric key generation and exchange technique (CK-SKGET). Based on an elaborate message matching and consistency verification mechanism, CK-SKGET uses historical communication messages to generate consistent common knowledge among communication peers. Then, CK-SKGET realizes secure key generation and exchange with low costs based on common knowledge. This paper designs a low-cost message matching mechanism without interfering or changing the normal communication mechanism. And we propose an algorithm for generating common knowledge through the historical communication messages of the communication peers. Furthermore, we provide a new idea for key exchange using communication messages. Thus, CK-SKGET has better security in key exchange and can be applied in broader network scenarios. Our contributions are summarized as follows.

1) A common knowledge generation and verification algorithm is designed. The communication peers use mutual communication messages to quickly establish common knowledge based on minimizing the number of additional information interactions and the risk of key leakage.

2) An algorithm for generating large prime numbers is designed which is based on the common knowledge established by the communication peers. Then, the communication peers use the DHKE protocol to realize the generation and exchange of symmetric keys, which improves the security in the process of symmetric key generation and exchange.

3) Through theoretical analysis and simulation comparisons, it was proved that CK-SKGET significantly improves the security of the symmetric key generation process under reasonable resource cost, also the feasibility and effectiveness of the scheme.

The rest of the paper is organized as follows. Section 2 introduces the system architecture, including the definition of common knowledge and system model. Section 3 introduces the generative process of common knowledge and the verification algorithm of common knowledge. Then, we use the common knowledge to compute large prime numbers and obtain symmetric keys with the DHKE protocol in Section 4. The security analysis of the proposed scheme is given in Section 5. Our proposed scheme was compared with others in Section 6, with the conclusion of our proposed scheme outlined in Section 7.

2 System Architecture

This section will define the common knowledge in this paper, and specifically explain the basic ideas and system architecture for the communication peers to build common knowledge and generate symmetric keys under the existing communication mode.

2.1 Definition of Common Knowledge

Common knowledge: To meet the user's application requirements, the communication peers will continuously exchange information and send encapsulated data frames to each other. During the exchange of data frames, we will gradually establish a shared and exclusive shared historical message database for them. The communication peers can calculate and abstract these historical messages to obtain a consistent understanding of the communication process and content-common knowledge. Since the common knowledge has the characteristics of mutuality, consistency, and exclusivity of the communication peers, we can apply it to the security protection process such as key generation and exchange, to make the communication process more secure.

Specifically, according to the role of each field in the message, these fields can be divided into two categories: control fields M^c that guarantee the communication process, and content fields M^d that are exchanged in communication. The format and length of the control field are deterministic and consistent, which is convenient for unified processing. The format and length of the content field are affected by communication requirements, and have variability and differences, and need to be truncated or zero-filled, etc. After the communication peers calculate the data in the control field and the content field, they will establish common knowledge of communication process and content.

Among the historical messages exchanged during the communication process between the communication peers, those historical messages M_i $(i \in \mathbb{N}^+)$ shared by the communication peers after negotiation and confirmation are called effective messages. Each effective message is represented as M_i^e $(i \in \mathbb{N}^+)$, and each filed in M_i^e is represented as $m_{i,j}$ $(i, j \in \mathbb{N}^+)$. These fields have two types: control field and content field, depending on the specific communication protocol used. For example, in the Ethernet frame protocol, a message M_i consists of seven fields: lead code $m_{i,1}$, frame start $m_{i,2}$, destination address $m_{i,3}$, source address $m_{i,4}$, protocol type $m_{i,5}$, data packet $m_{i,6}$, and parity code $m_{i,7}$. $m_{i,1}$, $m_{i,2}$, $m_{i,3}$, $m_{i,4}$, $m_{i,5}$, and $m_{i,2}$ belong to control fields, while $m_{i,7}$ belongs to content fields.

Therefore, the computation of common knowledge is primarily dependent on long-term communication between communication peers (depending on the security level, it can be in minutes, hours, days, months, or even years), continuous communication history, and a small amount of negotiation. It is theoretically possible for an attacker to eavesdrop on the entire process and steal the common knowledge based on the complete information overheard. However, due to the implementation cost and difficulty, it is almost impossible to carry out long-term continuous eavesdropping on each group of communication peers in the entire network. Besides, its eavesdropping is also less covert and easier to detect. Therefore, it can be considered that common knowledge has the characteristics of mutuality, consistency, and exclusivity. We can apply it to the security protection process such as key generation and exchange to make the communication process between the communication peers more secure.

2.2 System Model

The system structure of CK-SKGET is shown in Figure 1. After the communication frequency between communication peers reaches a certain threshold F_0 $(F_0 > 0)$, H_1 can decide to negotiate with H_2 and start building common knowledge of the communication process. The negotiation content is a 7-tuple $< T_0, N_m, N_p, c, d, R_C, R_D >$, where T_0 is the start time of collecting messages, $N_m (N_m \in \mathbb{N}^+)$ is the number of messages collected, c $(c > 3, c \in \mathbb{N}^+)$ is the number of columns of the control matrix, d $(d \in \mathbb{N}^+)$ is the number of columns of the content matrix, R_C $(R_C \in \mathbb{N}^+)$ and R_D $(R_D \in \mathbb{N}^+)$ are random number. If the message matching fails for the first time, it means that the communication message sets collected by H_1 and H_2, $S_1 = \{M_{1,1}, M_{1,2}, \cdots, M_{1,N_m}\}$ and $S_2 = \{M_{2,1}, M_{2,2}, \cdots, M_{2,N_m}\}$, are inconsistent.

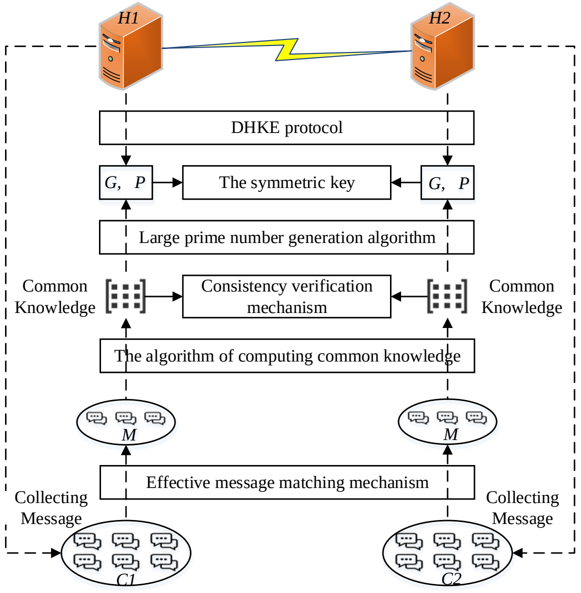
H1
H2
DHKE protocol
G, P
The symmetric key
G, P
Large prime number generation algorithm
Common Knowledge
Consistency verification mechanism
Common Knowledge
The algorithm of computing common knowledge
M
M
Effective message matching mechanism
Collecting Message
Collecting Message
C1
C2

Figure 1: The Structure of CK-SKGET

Then, based on the agreed rules, the communication peers take subsets s_i^1 and s_i^2 $(i \in \mathbb{N}^+)$ of sets S_1 and S_2, respectively. And they match s_i^1 and s_i^2, which is partial message matching. N_p is the maximum number of partial message matching.

The communication frequency F between H_1 and H_2 changes in real time. Suppose the time required to collect messages is T_m, the time for generating common knowledge and symmetric keys through historical messages is T_c, and the value of N_m is decided by $\psi(F, T_c)$. The design principle of the function $\psi(F, T_c)$ is as follows. 1) When F is small, N_m should be reduced to shorten the message collecting time, which aims to ensure that the key can be updated in time. 2) When F is large, N_m should be increased to make $T_m > T_c$ guaranteeing there is sufficient time to update the key. At the same time, the security of the key generated by CK-SKGET is related to the number of effective messages. When F is smaller, the minimum number of N_m is $N_0 = \psi(F, T_c)$ to ensure the security of the generated key. In order to update the key in time and improve the security of the key, H_1 and H_2 should negotiate the 7-tuple in each round of generating common knowledge. After H_1 and H_2 negotiate to generate symmetric key through CK-SKGET, they store the communication messages in their respective caches C_1 and C_2 starting from T_0.

When the message cached by H_1 reaches the predetermined number N_m, it sends an effective message matching request to H_2. They obtain an effective message set for generating common knowledge through an effective message matching mechanism M^v. Then, the communication peers use the common knowledge generation algorithm to obtain the common knowledge matrix K, and verify the consistency of K. Then, they use the eigenvalues of the matrix and the large prime number generation algorithm to get the same large prime number G and primitive root P. Finally, they execute the DHKE protocol to reach agreement of the symmetric key. The symmetric key generated by common knowledge will be used in the subsequent encryption and decryption of communication data to ensure the security of the data in the communication process.

3 Common Knowledge Generation and Verification Algorithm

The process of generating common knowledge must comply with the following basic principles: 1) H_1 and H_2 cannot interfere with the normal communication process and cannot change the mechanisms of the existing communication protocol. For example, operations such as retransmission and reassembly of lost and out-of-sequence data frames must be determined by the communication protocol, and no additional additions or deletions are permitted. 2) The messages in S_1 and S_2 are inconsistent, and it is necessary to make sure that the messages used to compute common knowledge are the same. 3) The key generation is based on common knowledge, and the common knowledge generated by the communication peers should be exactly the same. Based on the above principles, the establishment of common knowledge between H_1 and H_2 requires three steps: effective messages matching, computing common knowledge based on effective messages, and consistency verification of common knowledge.

3.1 Effective Messages Matching Mechanism

The effective messages matching mechanism is shown in Figure 2. It includes three stages: communication mode negotiation, communication messages collection and effective messages matching. In the stage of communication mode negotiation, H_1 and H_2 negotiate to start a new round of communication cycle and related information. H_1 sends the request of communication cycle start to H_2. H_2 responds to the request after receiving it. In the stage of communication messages collection, they carry out the normal communication interaction process. When the number of message buffers is N_1 , it will send effective messages matching requests. H_1 sends a complete message matching request to H_2, and after H_2 receives the request, they perform a complete message matching. If the match is successful, H_2 sends a response message to H_1, or they start partial message matching to obtain effective messages that can generate common knowledge. If they still fail to match after N_p times of partial message matching, they abandon the messages in this round of communication cycle and start a new round of communication cycle. After communication peers are successfully matched with effective messages, they obtain the set $M^e = \{M_1^e, M_2^e, \cdots, M_N^e\}$ of N effective messages, and send a matching success response to the other.

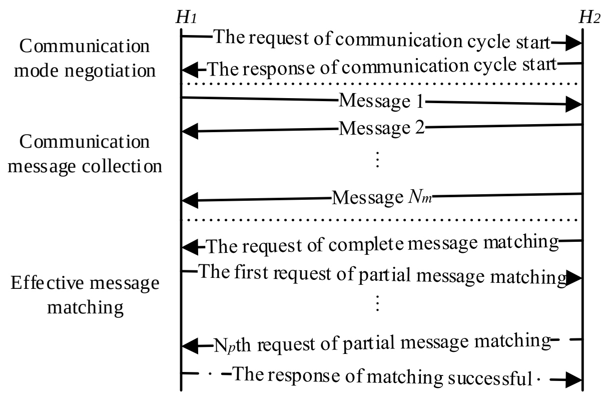
H1
H2
Communication mode negotiation
The request of communication cycle start
The response of communication cycle start
Communication message collection
Message 1
Message 2
Message Nm
Effective message matching
The request of complete message matching
The first request of partial message matching
Npth request of partial message matching
The response of matching successful

Figure 2: Effective message matching mechanism

When matching the complete message with the partial message, H_1 and H_2 will hash the message arranged in chronological order. When the hash results of the communication peers are consistent, the effective message is matched successfully. When performing partial message matching, H_1 and H_2 hash the set $s_i^1 = \{M_{1,1}, M_{1,2}, \cdots, M_{1,k}\}$ and $s_i^2 = \{M_{2,1}, M_{2,2}, \cdots, M_{2,k}\}$ $(k \in [1, N_m])$, respectively. The message $M_{1,j}$ and $M_{2,j}$ $(j \in [1, k])$ in the set s_i^1 and s_i^2, respectively, need to sample in S_1 and S_2 by function $S(x)$. The design principle of the function $S(x)$ is to ensure that in the expected N_p hashing result comparisons, there is as little overlap between the sampled messages as possible and the probability of a single message appearing in the two hashing comparison is as small as possible. In this paper, $S(x) = a \times i + 1$ is Step function, where $a = 2, 3, 4$ *et al.* The value of a is related to N_m and N_p. In actual applications, $S(x)$ is adjusted or redesigned according to the requirements of the communication scenario.

3.2 Computing Common Knowledge

To compute common knowledge, H_1 and H_2 firstly sample data from M_i^c and M_i^d in M_i^e to get the control matrix U and the content matrix V, respectively. Then the common knowledge matrix can be computed based on U and V.

In the construction matrix U and V, H_1 and H_2 obtain a row vector $\vec{u} = [u_1, u_2, \cdots, u_c]$ of U and $\vec{v} = [v_1, v_2, \cdots, v_d]$ of V by sampling M_i^c and M_i^d, respectively. When they sample the data in M_i^c, the bytes $b_{i,j}$ of M_i^c containing the fields are first sequentially concatenated into a field having l_i^c bytes. Then, they sample data from c, R_C, and l_i^c. The rules of sampling data are as follows.

1) If $l_i^c < c$,

$$u_y = \begin{cases} b_{i,j}^c, & j \leq l_i^c; \\ 0, & j > l_i^c. \end{cases} \quad (1)$$

2) If $l_i^c \geq c$, H_1 and H_2 divide the content of $b_{i,j}$ to c groups, and sample data from each group. The data size of each group is $\xi_i = \lfloor \frac{l_i^c}{c} \rfloor$, where "$\lfloor \bullet \rfloor$" is round-down operation. The initial value of R_i^s is R_C, and H_1 and H_2 adjust R_i^s according to ξ_i.

$$R_i^s = \begin{cases} 1 & , \xi_i = 1; \\ \lfloor \frac{R_i^s}{\xi_i} \rfloor & , R_i^s \geq \xi_i. \end{cases} \quad (2)$$

So that H_1 and H_2 obtain $u_y = b_{i,j}^c$ $(1 \leq y \leq c)$, $j \bmod \xi_i = R_i^s$ and $j \leq c \times \xi_i$. When H_1 and H_2 sample data in M_i^d, they replace d, R_D, l_i^d with c, R_C, l_i^c, respectively. Then, they obtain each data $v_y = b_{i,j}^d$ $(1 \leq y \leq d)$ of $\vec{v}$.

For the above sampling rules, the bytes in the message field are all regarded as sampled byte data when the number of bytes in the message field is less than the number of samples, and the rest is filled with zeros. When the number of bytes in the message field can support random sampling, the value of the sampling random number will be readjusted according to the number of groups to enable uniform data sampling from the message field. So that H_1 and H_2 construct a matrix U and a matrix V.

$$U = \begin{bmatrix} u_{1,1} & \cdots & u_{1,c} \\ \vdots & \ddots & \vdots \\ u_{N,1} & \cdots & u_{N,c} \end{bmatrix}, V = \begin{bmatrix} v_{1,1} & \cdots & v_{1,d} \\ \vdots & \ddots & \vdots \\ v_{N,1} & \cdots & v_{N,d} \end{bmatrix}$$

Then, on the basis of U and V, matrix W is

$$W = U^T \times V = \begin{bmatrix} w_{1,1} & \cdots & w_{1,d} \\ \vdots & \ddots & \vdots \\ w_{c,1} & \cdots & w_{c,d} \end{bmatrix} \quad (3)$$

The meaning of the matrix W is to project the content information into the space corresponding to the control information, so that H_1 and H_2 can understand the content information field in the same way as the control information field. But W is not a square matrix, so we can't find the eigenvalues directly. Multiply W with its transposed matrix W^T to get the common knowledge matrix of square matrix.

$$K = W \times W^T = \begin{bmatrix} k_{1,1} & \cdots & k_{1,c} \\ \vdots & \ddots & \vdots \\ k_{c,1} & \cdots & k_{c,c} \end{bmatrix} \quad (4)$$

After solving the eigenvalues and eigenvectors of the matrix K, a set $\lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_c\}$ of eigenvalues and a matrix $\gamma = \{\vec{\gamma_1}, \vec{\gamma_2}, \cdots, \vec{\gamma_c}\}$ of corresponding eigenvectors can be obtained. The computing process of H_1 and H_2's common knowledge is shown in Algorithm 1.

3.3 Consistency Verification of Common Knowledge

The matrix K is a real symmetric matrix according to (4), and the eigenvectors corresponding to different eigenvalues are orthogonal to each other. This paper uses the

Algorithm 1 Common Knowledge Generation

```
Input: $T_0, N_m, N_p, c, d, R_C, R_D$
Output: $K, \lambda, \gamma$
1: Effective messages $M^e \leftarrow$ Effective message matching
   mechanism$(T_0, N_m, N_p)$.
2: for Message $M_i^e$ in $M_e$ do
3:     $\vec{u}_i, \vec{v}_i \leftarrow$ Sampling $M_i^e$ by $c, d, R_C, R_D$.
4: end for
5: $W = U^T \times V$.
6: $K = W \times W^T$.
7: $\lambda, \gamma \leftarrow$ Decompose the eigenvalues of the matrix $K$.
8: End
```

eigenvectors of K to verify the consistency of the matrix calculated by H_1 and H_2. However, the eigenvector does not participate in the calculation process of large prime number, and the eigenvalue cannot be deduced from the eigenvector. So that transferring the characteristic vector in the communication network will not affect the security of the key agreement process.

The consistency verification process of common knowledge includes eigenvector multiplication verification and eigenvector hash verification. Before H_1 sends the consistency verification information of common knowledge, it selects n eigenvectors from γ to make up verification matrix $\eta_1 = [\vec{\gamma_1'}, \vec{\gamma_2'}, \cdots, \vec{\gamma_n'}]^T$, and sorts the remaining $c - n$ eigenvectors according to the size of the corresponding eigenvalues to form a matrix γ_1. η_1 and $hash_{\gamma_1}$ constitute the consistency verification information of common knowledge. After receiving the consistency verification information, H_2 uses γ to multiply η_1 to verify it. And the matrix ρ can be obtained by multiplying with η_1 and γ.

$$\begin{aligned} \rho &= \gamma \times \eta_1 \qquad (5) \\ &= \begin{bmatrix} \gamma_{1,1} & \cdots & \gamma_{1,c} \\ \vdots & \ddots & \vdots \\ \gamma_{c,1} & \cdots & \gamma_{c,c} \end{bmatrix} \times \begin{bmatrix} \gamma'_{1,1} & \cdots & \gamma'_{n,1} \\ \vdots & \ddots & \vdots \\ \gamma'_{1,c} & \cdots & \gamma'_{n,c} \end{bmatrix} \end{aligned}$$

Since the eigenvectors corresponding to different eigenvalues are orthogonal to each other, there are only n non-zero values in ρ and each column has only one non-zero value. When $\rho_{i,j} \neq 0$ $(i, j \in [1, c], \mathbb{N}^+)$, $\rho_{i,j} = \vec{\gamma_i} \times \vec{\gamma_j'} = 1$, and $\vec{\gamma_i} = \vec{\gamma_j'}^T$, H_2 verifies η_1 by non-zero value in $\rho_{i,j}$ and obtain the n feature vectors making up η_1. Then H_2 gets a matrix γ_1' composed of $c-n$ eigenvectors, and hash γ_1' to obtain $hash_{\gamma_1'}$, after which it compares $hash_{\gamma_1}$ and $hash_{\gamma_1'}$ to verify whether the remaining $c - n$ feature vectors are the same. If the above conditions are met, H_2 can determine that the common knowledge generated with H_1 is the same, and at the same time prevent a third party from tampering with the verification information of the common knowledge. Next, H_2 samples n' $(n' \in [1, c - n])$ eigenvectors from γ_1 to get verification matrix η_2. And H_2 obtains matrix γ_2 based on $c - n'$ eigenvectors in γ to obtain the consistency verification information of common knowledge of H_2 to H_1. H_1 verifies this verification information as H_2. The consistency verification process of the common knowledge using the verification message (including γ, η and $hash_{\gamma_x}(x \in 1,2)$) is shown in Algorithm 2.

Algorithm 2 Consistency Verification of Common Knowledge

```
Input: $\gamma, \eta, hash_{\gamma_x}$
Output: $True$ or $False$
 1: $\rho = \gamma \times \eta$.
 2: for $i = 1 \to c, j = 1 \to n$ do
 3:     if $\rho_(i,j) \neq 0$ and $\vec{\gamma_i} \neq \vec{\gamma}_j^{\,T}$ then
 4:         return $False$.
 5:     end if
 6: end for
 7: $\gamma^{'} = \gamma - \eta$.
 8: if $Hash(\gamma^{'}) \neq hash_{\gamma_x}$ then
 9:     return $False$.
10: end if
11: return $True$.
12: End
```

4 Key Agreement

This section mainly introduces the algorithm of using common knowledge to compute large prime numbers and the combination of DHKE protocol to obtain symmetric keys. The large prime number generation algorithm includes two parts: factor base update and large prime number computation.

4.1 Factor Base Update

The Miller-Rabin prime number detection algorithm is a widely used plasticity detection algorithm, which is based on the Fermat theorem. The Fermat theorem is as follows: there are prime number n and integer a, which satisfy $\gcd(a,n) = 1$, so $a^{n-1} \equiv 1(\mod n)$ [22]. The Miller-Rabin prime number detection algorithm is derived from Fermat theorem. Since n is a prime number, $n = 2^s \times r + 1$, where $r = 2 \times k + 1$ ($k \in \mathbb{N}^+$). For integer a, $\gcd(a,n) = 1$, so $a^r \equiv 1(\mod n)$ or $a^{2\times j \times r} \equiv -1(\mod n)$ with $0 \leq j \leq s-1$. The Miller-Rabin prime number detection algorithm is a probability detection algorithm, which means some strong pseudo prime numbers may be detected incorrectly [18]. After k times of testing, the probability of being wrongly judged as a composite number is $(\frac{1}{4})^k$. The speed of this prime number detection algorithm is much higher than that of other detection algorithms (such as Solovay-Strassen detection algorithm) [14].

This paper combines the rapid generation algorithm of large prime numbers from small prime numbers in [27] and the incremental prime number generation algorithm in [18] to generate odd numbers. And we use the Miller-Rabin prime number detection algorithm for primality detection to realize symmetric key generation based on common knowledge.

Not all elements in the set of eigenvalues λ are prime number. When constructing the small prime number set based on the set λ, we replace the odd number closest to each λ_i ($i \in [1, c]$, $\mathbb{N}^+$). Then we use the Miller-Rabin algorithm to detect the primality of λ_i. If λ_i is not the prime number, do $\lambda_i = \lambda_i + 2$ until λ_i is a prime number. Finally, after arranging λ_i in the set of small prime numbers by value, the factor base is obtained $B = \{b_1, b_2, \cdots, b_c\}(b_i = \lambda_i)$.

Suppose the factor base of the i-th round update is B_i, B_0 is the initial factor base. The prime number of B_i is $b_{i,j}$ ($j \in [1, c]$, $\mathbb{N}^+$). The formula of updating $b_{i,j}$ is as follows.

$$b_{i+1,j} = \prod_{k=1,k\neq c+1-j}^{c} b_{i,j}^{\alpha} + 2 \times \delta. \qquad (6)$$

α is the power factor, through which we can adjust the speed of updating B, and $\alpha = 1$ in this paper. δ is the distance between $b_{i+1,j}$ and the odd number $\prod_{k=1,k\neq c+1-j}^{c} b_{i,j}^{\alpha}$ obtained by multiplying multiple prime factors, the initial value of which is 0. If the Miller-Rabin algorithm detects that $b_{i+1,j}$ computed from Formula (6) is not prime, do $\delta = \delta + 1$ to obtain new $b_{i+1,j}$ until $b_{i+1,j}$ is a prime number. This process is called incremental prime number generation algorithm. After computing the prime number $b_{i+1,c}$, we can obtain B_{i+1}.

Before updating B_i, it will first compare the length $f(b_{i+1,j})$ of the largest prime $b_{i+1,j}$ in B_{i+1} with the length $\tilde{P}$ of the prime P. So that we can judge whether the factor base B_i is the last round of factor base. The function $f(x)$ for estimating the number of prime numbers $b_{i,j}$ is as follows.

$$f(x) = \lceil \log_2 b_{i,j} \rceil \qquad (7)$$

"$\lceil \bullet \rceil$" is round-up operation, such as $f(50) = \lceil \bullet \rceil = 6$. Suppose the product of two integers a_0 and a_1 is a_2, so $\log_2 a_2 = \log_2 a_1 + \log_2 a_0$, and the length of a_2 satisfies

$$f(a_0) + f(a_1) - 1 \leq f(a_2) \leq f(a_0) + f(a_1). \qquad (8)$$

Put the above formula into Formula (6),

$$\sum_{k=1,k\neq c+1-j}^{c} f(b_{i,k}) - c + 1 \leq f(b_{i+1,j}) \leq \sum_{k=1,k\neq c+1-j}^{c} f(b_{i,k}). \qquad (9)$$

The number of prime numbers calculated in the next round is less than $\tilde{P}$. When $\tilde{P} > f(b_{i+1,c})$, shown B_i is not the last round factor base B_l. So we need compute B_{i+1} and retain $b_{i,c}$ to add it to the set ϕ. If not, $\tilde{P} < f(b_{i,c})$, shown $B_l = B_i$, we stop updating B_{i+1}.

The number of length of $b_{i+1,j}$ depends on the prime factors involved in the process of calculating it according Formula (9), and the length difference between $b_{i+1,j}$ and $b_{i,j}$ is huge. The differences in the lengths of prime numbers in B_0 will affect the prime length differences after the factor base is updated. When the difference in lengths between $b_{0,1}$ and $b_{0,c}$ are little, so the difference in lengths between $b_{1,1}$ and $b_{1,c}$ are also little after the factor base is updated. But the length difference between $b_{1,j}$ and $b_{0,j}$ is huge. Thus $f(b_{i,j}) \approx f(b_{i,j+1})$ and $f(b_{i+1,j}) \approx (c-1) \times f(b_{i,j})$. So that the update end condition of updating B_i is to determine whether $f(b_{i+1,c})$ is greater than $\tilde{P}$. Meanwhile, $b_{l,j}$ in B_l is not fully in involved the compute process of P, so the update speed of the factor base can be improved by reducing the number of calculations of $b_{l,j}$. Because of $\theta_l = \lfloor \frac{\tilde{P}}{f(b_{l,c})} \rfloor$ and $\theta \in \mathbb{N}^+$, θ_l is the number of participating in the compute process of $\tilde{P}$, we can obtain the primes numbers $\{b_{l,c-\theta_l-1}, \cdots, b_{l,c}\}$ in B_l.

Due to the uneven distribution of prime numbers, in the process of computing the prime number $b_{i+1,j}$, the number of prime number determinations with Miller-Rabin algorithm is random [18]. But the time of computing $b_{i+1,j}$ is increasing with the length of $b_{i+1,j}$. Therefore, it takes a certain amount of time to update B_i, and the subsequent prime numbers cannot be calculated directly from the initial factor base B_0, preventing a third party from quickly cracking the prime numbers in the key agreement process.

4.2 Large Prime Number Computation

After updated the factor base, the set $\{b_{l,c-\theta_l}, \cdots, b_{l,c}\}$ in B_l and the set $\phi = \{b_{0,c}, b_{1,c}, \cdots, b_{l-1,c}\}$ make up the direct factor base B_ϕ to compute P. Because of

$$L_1 = \sum_{j=\theta_l}^{c} f(b_{l,j}) \leq \tilde{P} < \sum_{j=\theta_l-1}^{c} f(b_{l,j}) = L_2, \qquad (10)$$

when b_{l,θ_l-1} is involved in the compute of prime numbers, the length of computed odd number will be greater than P. We only need to compute θ_l prime numbers.

The number of $b_{i,c}$ in B_ϕ is θ_i ($\theta_i \in \mathbb{N}$).Firstly, due to $\Delta L = \tilde{P} - L_1$, $\Delta L - \theta_i \times f(b_{i,c}) \leq f(b_{i,c})$, the number of $b_{l-1,c}$ is θ_i. Then, $\Delta L = \Delta L - \theta_i \times f(b_{i,c})$, we repeat this process to get the set

$$\Phi = \{\{b_{l,\theta_l}, \cdots, b_{l,c}\}^{\theta_l}, \{b_{0,c}, \cdots, b_{0,c}\}^{\theta_0}, \cdots, \{b_{l-1,c}, \cdots, b_{l-1,c}\}^{\theta_{l-1}}\}. \qquad (11)$$

Through

$$P = 2^\beta \times \prod_{i=0}^{l-1} b_{i,c}^{\theta_i} \times \prod_{j=\theta_l}^{c} b_{l,j} + 2 \times \delta, \qquad (12)$$

we can compute P. However, since the length of $P_d = \prod_{i=0}^{l-1} b_{i,c}^{\theta_i} \times \prod_{j=\theta_l}^{c} b_{l,j}$ and $\tilde{P}$ are not necessarily equal, we multiply it with 2^β ($\beta \in \mathbb{N}^+$) to get an odd number whose lengths is $\tilde{P}$. Through $\beta = \tilde{P} - f(P_d)$, we obtain the prime number P by the incremental prime number generation algorithm.

4.3 Key Information Exchange

The basic idea of the DHKE protocol is to take advantage of the difficulty in computing the discrete logarithm. The communication parties send only part of the data generated by the key and retain the other part of the data, respectively, to achieve key security agreement [11]. After H_1 and H_2 generate the large prime numbers P and G using their common knowledge, they negotiate the key using the DH key exchange protocol as follows:

1) H_1 generates a random number R_1 ($R_1 \in \mathbb{N}^+$), then according to the following exchange information it can be calculated the key exchange information $X_1 = G^{R_1} \mod P$ which can be transmitted in the public channel, and transmit X_1 to the H_2.

2) H_2 generates a random number R_2 ($R_2 \in \mathbb{N}^+$), after receiving the key exchange message from H_1, and transmits X_2 and to H_1, then the symmetric key is calculated, where $Key = X_1^{X_2} \mod P$.

3) H_2 calculates the symmetric key, where $Key = X_2^{X_1} \mod P$.

From $Key_1 = Key_2 = G^{R_1 \times R_2} \mod P$, H_1 and H_2 can get the same key. Decisional Diff-Hellman (DDH) hypothesis states that it is difficult to distinguish the tuples (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, g^z), among which g is a generator and $\{x, y, z\}$ is a set of random numbers; when G and P are very large, and R_1 and R_2 cannot be obtained at the same time, it is difficult to calculate the shared key, though the DHKE protocol cannot resist MITM attacks. In the process of symmetric key negotiation, the common knowledge matrix K cannot only generate shared large primes against the MITM attacks, but also provide identity authentication for key information exchange which guarantees the security of the whole key generation process.

5 Security Analysis

This section analyzes the security of the proposed scheme in theory and compares the security performance of the proposed scheme with the existing mechanism in combination with typical attack modes.

5.1 Security Model

Based on the Random Oracle model proposed in [6], a security analysis model for CK-SKGET scenario is designed. Suppose the legal participants are H_1 and H_2, and an probabilistic polynomial time (PPT) enemy $\mathcal{A}$. Through inquiring the session instance between H_1 and H_2 ($\prod_{H_1}^n, \prod_{H_2}^n$), the enemy attempts to obtain the key. The specific model is as follows:

1) $Execute(\prod_{H_1}^n, \prod_{H_2}^n)$ query: simulating the passive attack launched by the adversary, $\mathcal{A}$ make communication between H_1 and H_2 through the query, and obtain all the contents in the communication process.

2) $Hash(M_i)$ query: H_1 or H_2 inquires the hash value, $Hash(M_i)$, which are corresponding to the message, M_i $(i \in \mathbb{N}^+)$ from $\mathcal{A}$. If the hash value exists in the list, L_H, it is returned to A; otherwise, a random number $R_i^H \in \mathbb{Z}_q^+$ is as the hash value of M_i returning to $\mathcal{A}$, and $M_i, R_i^H \in \mathbb{Z}_q^+$ are stored in L_H.

3) $Send(\prod_{H_1}^n / \prod_{H_2}^n, M_i)$ query: simulating an active enemy attack, $\mathcal{A}$ sends a random message M_i to H_1 or H_2, after receiving the message, according to the numerical procedure H_1 or H_2 will calculate and return the result to $\mathcal{A}$.

4) $Reveal(\prod_{H_1}^n / \prod_{H_2}^n)$ query: simulating an attack from the enemy knowing the session key, after H_1 or H_2 receiving the query, the key negotiated by the scheme is returned to $\mathcal{A}$.

5) $Knowledge(CK)$ query: simulating an attack from the enemy knowing the common knowledge, after H_1 or H_2 receiving the query, the large prime calculated by the scheme is returned to $\mathcal{A}$.

6) $Test(\prod_{H_1}^n / \prod_{H_2}^n)$ query: $\mathcal{A}$ selects H_1 or H_2 for the session test, if $\mathcal{A}$ calculates the key, then return $\perp$ and stop the query; otherwise, through a coin toss to determine the value returned to $\mathcal{A}$, if it is head, the real key is returned to $\mathcal{A}$; otherwise, return a random number as long as the real key. But $\mathcal{A}$ is only allowed for one coin toss.

Definition 1 (Security define of CK-SKGET). *For any adversary $\mathcal{A}$, event Succ means that the key can be obtained by $\mathcal{A}$ through the above query processes, i.e., the attack is successful; in CK-SKGET, the winning edge of $\mathcal{A}$ is $Adv_{\mathcal{A}} = |Pr(S) - 1/2|$; if the value of $Adv_{\mathcal{A}}$ is negligible, the key generated from common knowledge is secure.*

5.2 Security Proof

Theorem 1 (Security of CK-SKGET). *If H: $\{0,1\}^+ \to \{0,1\}^l$ is a random predictor, when the DHH hypothesis is true, the margin of $\mathcal{A}$ attacking DHH hypothesis is negligible, and the biggest margin of A to attack CK-SKGET is*

$$Adv_{CK-SKGET} \leq \frac{(q_e+q_s)^2}{2^{l_c}} + \frac{{q_h}^2}{2^{l_c}} + \frac{{q_s}^2}{2^{l_g+l_p}} + q_h Adv_{DDH}, \quad (13)$$

where the maximum number of times that $\mathcal{A}$ can initiate Hash query, Execute and Send query are q_h, q_e and q_s respectively, and the bit lengths of consistency verification information, large prime G and P outputted by hash function are l_c, l_g and l_p respectively.

Proof. the security of CK-SKGET is verified by a series of games(G) among participants and enemy. G includes five games from $Game_0$ to $Game_4$, define $Succ_i$ represents $\mathcal{A}$ wins $Game_i$, and the ability of $\mathcal{A}$ increases gradually according to the query process in 5.1.

$Game_0$: The game is a real scene, and known from **Defintion 1**.

$$Adv_{CK-SKGET} = |\Pr(S) - 1/2| \quad (14)$$

$Game_1$: based on $Game_0$, $\mathcal{A}$ can steal the communication between H_1 and H_2. H_1 and H_2 run Plan $\mathcal{P}$, and maintain the list $L_H = \{M, h\}$, which is used to store the query and answer to $\mathcal{A}$. It is known that when $\mathcal{A}$ can get the content of communication history, its margin is still equal to the one of attacking the DHH security hypothes which is negligible, i.e., $Game_0$ and $Game_1$ are indistinguishable in the predictor,

$$\Pr(Succ_1) - \Pr(Succ_0) = 0 \quad (15)$$

$Game_2$: based on $Game_1$, $\mathcal{A}$ can send message to H_1 or H_2 actively to form the content of communication history. But $\mathcal{A}$ does not know the generation mechanism of common knowledge, then consistency verification information can only be formed through hash collision, which terminates when hash collision occurs. From the birthday paradox, the probability of collision in the predictor is $\max(q_h/2^{l_c})$; if $\mathcal{A}$ cannot obtain the content of communication history, the collision probability is $\max[(q_e+q_s)^2/2^{l_c}]$, then

$$\Pr(Succ_2) - \Pr(Succ_1) \leq \frac{(q_e+q_s)^2}{2^{l_c}} + \frac{{q_h}^2}{2^{l_c}} \quad (16)$$

$Game_3$: based on $Game_2$, $\mathcal{A}$ conjectures the large prime number G and P through the common knowledge, then the margin of A winning the game is

$$\Pr(Succ_3) - \Pr(Succ_2) \leq \frac{{q_s}^2}{2^{l_c+l_p}} \quad (17)$$

$Game_4$: during the game, $\mathcal{A}$ can only guess the key by the predictor attempting to solve the problem of DHH, where the margin is

$$\Pr(Succ_4) - \Pr(Succ_3) = q_h Adv_{DHH} \quad (18)$$

From the five games above, after casting all attacks, the margin of $\mathcal{A}$ under the security model described in Section 5.1 is

$$Adv_{CK-SKGET} \leq \frac{(q_e+q_s)^2}{2^{l_c}} + \frac{{q_h}^2}{2^{l_c}} + \frac{{q_s}^2}{2^{l_g+l_p}} + q_h Adv_{DDH} \quad (19)$$

□

5.3 Typical Attack Analysis

5.3.1 Typical Attack Modes

On the basis of security proof, the security attributes of CK-SKGET, such as bidirectional authentication, anti-replay attack, anti-password guessing attack and forward security, are further analyzed.

1) Bidirectional authentication. Based on the common knowledge established, only when the generation method of common knowledge matrix K is mastered, can H_1 and H_2 pass feature vector multiplication verification and feature vector hash verification which is used to verify the identities of the two in the consistency verification phase of shared knowledge.

2) Anti-replay attack. If $\mathcal{A}$ sent message to H_1 and H_2 and cannot pass the consistency verification of shared knowledge since the lack of calculation method of matrix K, $H_1$and H_2 are able to detect the presence of $\mathcal{A}$ and stop key negotiation.

3) Anti-password guessing attack. If enemy $\mathcal{A}$ send randomly guessed key exchange messages to H_1 and H_2, as the digits and the number of large prime numbers increase, the probability of $\mathcal{A}$ using the guessed large prime number for key negotiation is extremely low.

4) Forward security. As the communication interaction between H_1 and H_2 continues, the key can be continuously generated for communication encryption and decryption. Therefore, even if $\mathcal{A}$ can obtain the key of the current communication process through other methods, it is still difficult to obtain the correct large prime number and the negotiated key without CK-SKGET.

5.3.2 Security Comparison

This section compares several schemes related to the proposed scheme from five security attributes, such as bidirectional authentication and anti-replay attack, and it shows that CK-SKGET generates symmetric keys with better security. The results are shown in Table 1.

In [3], two-level private keys are designed to improve the DH key exchange protocol and hash the final negotiated key at the same time, but the identity of H_1 and H_2 cannot be bidirectional authentication. In [16], the server sends large prime numbers to H_1 and H_2, uses asymmetric encryption to ensure the security of data transmission, bidirectional identity authentication of H_1 and H_2 and anti-MITM attack, and Geffe binary generator can resist password guessing attack. However, H_1 and H_2 cannot resist the replay attack by the enemy by sharing large prime information, nor can they guarantee the forward security of the communication process. In [26], the authors can carry out bidirectional authentication on H_1 and H_2 by using the shared secret matrix to resist password guessing attacks and MITM attacks, but cannot resist replay attacks by using previously used eigenvalues. CK-SKGET implements bidirectional identity authentication for H_1 and H_2 and resolves the MITM attacks, anti-key exchange information replay attack, anti-password guessing attack, and forward security in the DH key exchange protocol.

6 Simulation Results and Performance Analysis

This section conducts simulation verification for CK-SKGET to evaluate its performance. The simulated physical device is H_1 and H_2 connected through an Ethernet, and the data transmission rate is set to 100 Mb/s. The communication behavior per second between H_1 and H_2 obeys the Poisson distribution. The application layer data length in the Ethernet frame sent obeys the uniform distribution over the interval $[46, 1500]$. Each byte sent in communication data obeys the uniform distribution over the interval $[1, 255]$. The language used in the simulation in this paper is Python. Firstly, analyze the time to crack CK-SKGET, the traditional DHKE protocol and the RSA key exchange protocol. And to effectively evaluate the CK-SKGET, the storage resource cost and the calculation time cost caused by the key generation process of the three schemes are compared through the average analysis of 100 experiments on the PyCharm platform.

6.1 Creaking Time Analysis

In CK-SKGET, the guarantee of key security is provided by the DH key agreement mechanism and the process of generating shared large prime numbers. The security of the DHKE protocol is a discrete logarithm problem, which is determined by the number of large prime numbers in the key exchange information [11]. The security of the shared large prime number generation process is guaranteed by common knowledge. If an adversary tries to obtain the communication key of H_1 and H_2 by creaking the shared large prime number, it is necessary to obtain the shared knowledge matrix from the effective message when the adversary want to pass the consensus verification process of the common knowledge and the message verification code process during key information exchange. Then the key agreement process between H_1 and H_2 can be successfully attacked. Therefore, the adversary mainly cracks the common knowledge from the effective message and then cracks the CK-SKGET.

The security of the traditional DHKE protocol and the RSA key exchange protocol is related to the key transmission mechanism and the key. The key transmission mechanism of the two protocol guarantee the security based on the mathematical problem principle. For the security of the key, the DHKE protocol is determined by the shared large prime number. However, the generation mechanism of shared large prime numbers is the same as the key generation mechanism of the RSA key exchange protocol. The random number generator gives a random number that meets the requirements of the number of key lengths, and then obtains the prime number through the prime number generation algorithm and the prime number judgment algorithm. Therefore, the security of the key in the traditional key exchange protocol is mainly determined by the number of prime numbers [5].

Suppose that the adversary can obtain the effective

Table 1: Comparison of security attributes

Lterature	[3]	[16]	[26]	CK-SKGET
Two-way authentication	N	Y	Y	Y
Anti-replay attack	Y	N	N	Y
Anti-password guessing attack	Y	Y	Y	Y
Forward safety	Y	N	Y	Y
Anti-MITM attacks	Y	Y	Y	Y

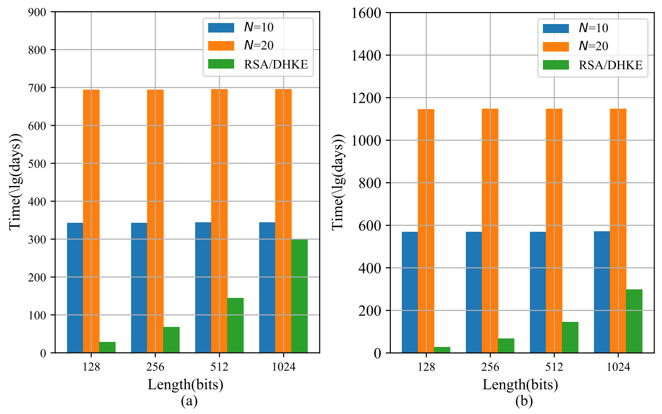

Figure 3: Comparison of time cost for creaking large prime number. (**a**) The time of cracking large prime number when $d = 15$. (**b**) The time of cracking large prime number when $d = 30$.

message between the sink node and the cloud, and it is clear that the process of obtaining matrix K from matrix U and V and computing shared large prime numbers. The time cost required for the adversary to crack the scheme is analyzed. When the adversary cracks the CK-SKGET, it constantly tries to obtain the vector $\vec{u}$ of length c and the vector $\vec{v}$ of length d from each valid message, and then generate matrix U and V. The number of counts the adversary crack CK-SKGET is mainly related to the number of effective messages N and the columns d of the content matrix. The lengths of prime number P to be found is the abscissa and the logarithm of the cracking time is the ordinate. When $c = 5$, we respectively get CK-SKGET in the case of $d = 15$ and $d = 30$, the number of effective messages and the key cracking time cost of the traditional key exchange method, as shown in Figure 3.

When cracking CK-SKGET, the adversary is mainly trying to obtain different matrix U and V, and the number of lengths of large prime numbers only affects its final calculation of large prime numbers. For traditional methods, the number of prime numbers directly affect the time taking for the adversary to crack the key. Therefore, in Figure 3, when N is unchanged, the cracking time corresponding to different $\tilde{P}$ is basically unchanged. But different N and d will seriously affect cracking time. The number of primes is larger, the time of cracking key is longer for traditional methods. Therefore, when H_1 and H_2 use CK-SKGET for key agreement, increasing the number of valid messages and the number of content matrix columns can improve the security.

6.2 Storage Resource Cost Analysis

In the process of generating common knowledge, the matrix U and V will take up a lot of space, it's about $m \times (c+d)$ bytes. In the process of prime number generation, the factor base updated in the last-round will occupy a large space and is related to the number of digits of each prime factor. Then the updated average number of prime factors can be known by Formula (9), which is

$$(L_0 - z) \times (c-1)^z \le f(\bar{b}_i) \le L_0 \times (c-1)^z. \qquad (20)$$

L_0 is the average length of the initial factor base, z is the number of update rounds of the factor base. Analyzing the calculation process of matrix K, we can get $L_0 = f(\bar{\lambda}_i) \le \lceil \log_2 r \times N^2 \times \tau^4 \rceil$, where τ is the average value of each byte in the message. The number of messages has little effect on the storage resource consumption during prime number generation. We use the length of the prime number (bits) as the abscissa and the storage resource overhead as the ordinate. When $c = 15$ and $d = 15$, we respectively draw the matrix U and V, the storage resource cost of prime number generation and the traditional method, as shown in Figure 4.

In Figure 4(a), when the number of messages and $\tilde{P}$ are small, the storage resources occupied by the attribute matrix are greater than the prime number generation process. When $\tilde{P} = 512$ or $\tilde{P} = 1024$, the storage resource of the prime number generation process is larger than the attribute matrix. In Figure 4(b), when the number of messages is larger, the space occupied by the attribute matrix will always be greater than the prime number generation process. However, the storage resource cost of CK-SKGET will always be greater than the storage resources occupied by the prime number generation process used in the traditional DHKE protocol and the RSA key exchange protocol. The storage resource occupied by CK-SKGET is maintained at 1-100 Kb, and the increased storage overhead is still acceptable compared to the storage capacity of current smart devices.

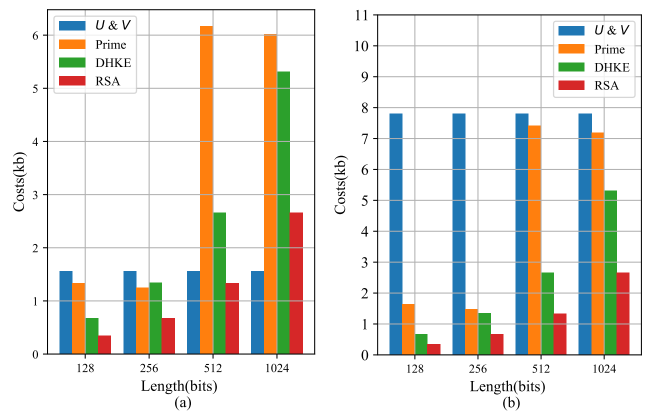

Figure 4: Comparison of storage resource cost. (**a**) The cost of storage resource when the number of messages is $m = 10$. (**b**) The cost of storage resource when the number of messages is $m = 50$.

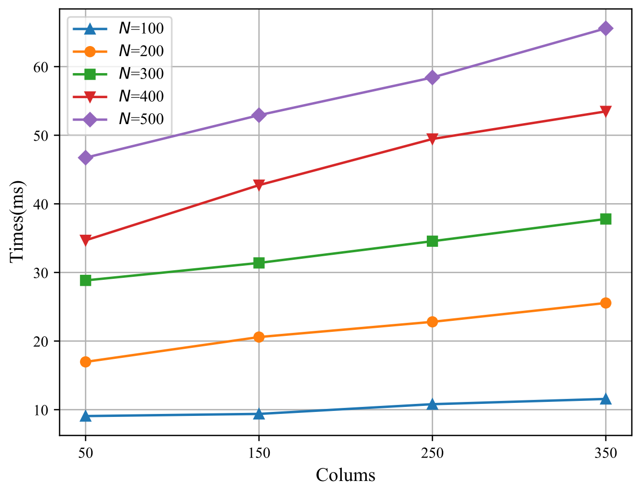

Figure 5: Comparison of computing cost for generating common knowledge

6.3 Computing Cost Analysis

The computing cost of CK-SKGET is mainly reflected in the process of common knowledge generation and the key generation based on common knowledge. The main computing cost of the latter comes from the large prime number generation process and the key agreement process. And the computing cost of the key agreement process is less than that of the large prime number generation process. Therefore, the computing cost of CK-SKGET mainly consumes time in the process of calculating common knowledge and computing large prime numbers.

6.3.1 Cost Analysis of Computing Common Knowledge

In the common knowledge generation stage, the main computing cost includes matrix operations and the eigenvalue decomposition of the common knowledge matrix. The computing cost of the former is related to effective messages N and columns of the content matrix d, but the computing cost of the latter is related to columns of the control matrix c. We take d as the abscissa and the computing cost of common knowledge as the ordinate. In the case of $c = 5$, we draw the computing cost curves under different situations, respectively, in Figure 5.

In Figure 5, when N is smaller, increasing d has a small impact on the time of the common knowledge generation process, and the computing cost of curves corresponding to different N is larger. But the value of N is larger, the impact on computing cost is higher. Therefore, the computing cost of common knowledge generation can be kept at a stable level by adjusting the number of messages and the number of columns in the content matrix.

6.3.2 Cost Analysis of Generating the Prime Number

In the prime number computing stage, the time of searching large prime numbers is a random value. And the cost of computing prime numbers increases with the length of prime numbers. The prime numbers in the factor base are better to compute than the large prime numbers sought. When changing the value of c, we can obtain a new number of rounds updating B since the count of eigenvalues is decided by the control matrix columns. After updated B, we compute the direct prime factor of the large prime number from the method in Section 4.2. Then we will compute P_d and find the prime number P closest to P_d. The time consumed in this process is related to the number of prime numbers, which is described in detail in the [18]. The two traditional key exchange methods to generate prime numbers are the same. So we compared the computing cost of CK-SKGET and traditional key exchange methods to generate prime numbers. And We use the number of prime numbers to be sought as the abscissa, and the time spent in computing large prime numbers under different prime factors is the ordinate, as shown in Figure 6.

In Figure 6, the cost of the prime number generation process is mainly related to the number of prime numbers to be sought and the number of prime factors (the number of columns c in the control matrix). From formula 20, we know that the rounds of updating B_i is related to the lengths $f(b_{i,j})$ of the prime number in B_i when seeking prime numbers with different lengths. The $f(b_{i,j})$ is larger, the rounds is more and the cost of time is longer. For example, when $\tilde{P} = 512(bits)$, $f(b_{i,j}) \leq 199(bits)$in the curve of $c = 6$, which is smaller than the curves of $c = 3, 4, 5$. But when $\tilde{P} = 1024(bits)$, $f(b_{i,j}) \leq 996(bits)$in the curve of $c = 6$, which is larger than the curves of $c = 3, 4, 5$, its cost of time is the largest. When the number of prime numbers is smaller, the com-

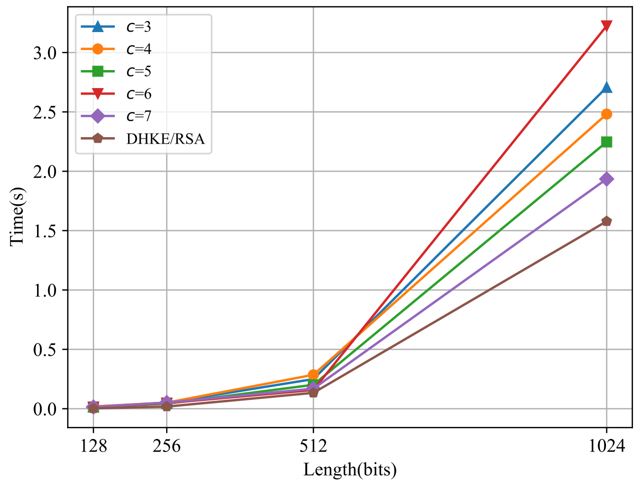

Figure 6: Comparison of computing cost for large prime numbers

puting cost of prime numbers is smaller. Therefore, the count of prime numbers in B_i base has a direct impact on the computing cost of prime number. Meanwhile, according to the prime number theorem [13], due to the larger interval between 512 bits and 1024 bits, the increase in computation time is larger than that between 256 bits and 512 bits [13]. Furthermore, the lowest computing cost of CK-SKGET is also close to that of the traditional key exchange method.

Comparing Figure 5 and Figure 6, when the number of prime numbers is larger, the computing cost of the common knowledge generation process is much smaller than the cost of the prime number generation process. So the cost of computing CK-SKGET mainly comes from the prime number generation process, considering that the prime number generation process also exists in the traditional key exchange method. The computing cost added by CK-SKGET only accounts for a very small proportion of the overall cost, which does not significantly increase the system computational burden, compared with the traditional key exchange method.

7 Conclusions

To resist the MITM attack in the key agreement process, this paper proposes a symmetric key generation and exchange technology based on common knowledge. Firstly, the communication peers obtain the same common knowledge through the common knowledge generation and verification algorithm. Then, they conduct symmetric key generation and exchange based on the common knowledge, and resist the MITM attacks in the key generation process. Theoretical analysis and simulation results show that CK-SKGET has theoretically provable security and can resist some typical attacks, thereby ensuring the security of the communication. Besides, compared with the traditional key agreement scheme, the storage cost and computing cost of CK-SKGET are not obvious. As for future work, the construction efficiency of common knowledge can be further improved, and the common knowledge can be extended to a wider range of application scenarios such as enhancing privacy protection, improving communication efficiency, and enabling tacit communication.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (U1705263), the Science Project of Yunnan Power Grid (YNKJXM20200172) and the Special Science Foundation of Quzhou (2021D013). The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] M. D. Abbasinezhad and M. Nikooghadam, "Efficient anonymous password-authenticated key exchange protocol to read isolated smart meters by utilization of extended chebyshev chaotic maps," *IEEE Transactions on Industrial Informatics.*, vol. 14, no. 11, pp. 4815–4828, 2018.

[2] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proceeding of 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 5–17, New York, NY, USA, 2015.

[3] H. M. Ahmed and R. W. Jassim, "Distributed transform encoder to improve diffie-hellman protocol for big message security," in *Proceedings of 3rd International Conference on Engineering Technology and its Applications*, pp. 84–88, IEEE, Iraq, 2020.

[4] S. Ali, A. Humaria, M. S. Ramzan, I. Khan, S. M. Saqlain, A. Ghani, J. Zakia, and B. A. Alzahrani, "An efficient cryptographic technique using modified diffie–hellman in wireless sensor networks," *International journal of distributed sensor networks*, vol. 16, no. 6, p. 1550147720925772, 2020.

[5] R. Alvarez, G. C. Caballero, J. Santonja, and A. Zamora, "Algorithms for lightweight key exchange," *Sensors.*, vol. 17, no. 7, pp. 15–17, 2017.

[6] M. Bellare and D. R. Pointcheval, "Authenticated key exchange secure against dictionary attacks," in *International conference on the theory and applications of cryptographic techniques*, pp. 139–155, Springer, 2000.

[7] T. Bui and T. Aura, "Key exchange with the help of a public ledger," in *Cambridge International Workshop on Security Protocols*, pp. 123–136, 2017.

[8] Y. Cao, N. Li, and J. Pan, "Key agreement protocol for dynamic identity authentication based on

chaotic mapping," *Mathematics in Practice and Theory*, vol. 51, no. 14, p. 9, 2021.

[9] C. Chunka, S. Banerjee, S. Nag, and R. S. Goswami. "A secure key agreement protocol for data communication in public network based on the diffie-hellman key agreement protocol,". in *Micro-Electronics and Telecommunication Engineering*, vol. 106, pp. 531–543, Singapore, 2020.

[10] Z. Dar, A. Ahmad, F. A. Khan, F. Zeshan, R. Iqbal, H. H. Sherazi, and A. K. Bashir, "A context-aware encryption protocol suite for edge computing-based iot devices," *The Journal of Supercomputing.*, vol. 76, no. 4, pp. 2548–2567, 2020.

[11] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory.*, vol. 22, no. 6, pp. 644–654, 1976.

[12] J. Y. Eun and S. J. Il, "An efficient and secure diffie–hellman key agreement protocol based on chebyshev chaotic map," *Communications in Nonlinear Science and Numerical Simulation*, vol. 16, no. 6, pp. 2383–2389, 2011.

[13] P. Hoffman, *The Man Who Loved Only Numbers*, vol. 45. New York: Hyperion Books, 1998.

[14] J. Hurd, "Verification of the miller–rabin probabilistic primality test," *The Journal of Logic and Algebraic Programming*, vol. 56, no. 1-2, pp. 3–21, 2003.

[15] S. Inshi, R. Chowdhury, M. Elarbi, s. H. Ould, and C. Talhi, "LCA-ABE: Lightweight context-aware encryption for android applications," in *International Symposium on Networks, Computers and Communications*, pp. 1–6, IEEE, 2020.

[16] A. S. Khader and D. Lai, "Preventing man-in-the-middle attack in diffie-hellman key exchange protocol," in *22nd International Conference on Telecommunications*, pp. 204–208, IEEE, 2015.

[17] H. Lai, M. A. Orgun, J. Xiao, J. Pieprzyk, L. Xue, and Y. Yang, "Provably secure three-party key agreement protocol using chebyshev chaotic maps in the standard model," *Nonlinear Dynamics.*, vol. 77, no. 4, pp. 1427–1439, 2014.

[18] F. Li, Z. Gong, F. Lei, S. Gu, and P. Gao, "Overview of fast prime numbers generation," *Journal of Cryptologic Research.*, vol. 6, no. 4, pp. 463–476, 2019.

[19] L. Liu and J. Cao, "Analysis of one key agreement scheme for bans based on physiological features," *International Journal of Electronics and Information Engineering*, vol. 13, no. 4, pp. 142–148, 2021.

[20] B. Majid and R. A. Mohammad, "An attribute based key agreement protocol resilient to kci attack," *International Journal of Electronics and Information Engineering*, vol. 2, no. 1, pp. 10–20, 2015.

[21] N. Naher and M. M. Haque, "Authentication of diffie-hellman protocol against man-in-the-middle attack using cryptographically secure crc," in *International Ethical Hacking Conference 2018*, vol. 811, pp. 139–150, Singapore, 2019.

[22] R. P. Sah, U. N. Roy, A. K. Sah, and S. K. Sourabh, "Early proofs of fermat's little theorem and applications," *International Journal of Mathematics Trends and Technology.*, vol. 64, no. 2, pp. 74–79, 2018.

[23] W. Shen, Y. Cheng, B. Yin, J. Du, and X. Cao, "Diffie-hellman in the air: A link layer approach for in-band wireless pairing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 11, pp. 11894–11907, 2021.

[24] W. Shen, W. Hong, X. Cao, B. Yin, D. Shila, and Y. Cheng, "Secure key establishment for device-to-device communications," in *IEEE Global Communications Conference*, pp. 336–340, Austin, USA, 2014.

[25] P. P. Thwe and M. Htet, "Prevention of man-in-the-middle attack in diffie-hellman key exchange algorithm using proposed hash function," *International Journal of Advances in Scientific Research and Engineering (IJASRE)*, vol. 5, p. 10, 2019.

[26] Y. Zhang, Z. Wang, Z. Wang, and H. Chen, "Verifiable three-party secure key exchange protocol based on eigenvalue," *Journal of communication.*, vol. 40, no. 12, pp. 149–154, 2019.

[27] L. Zhou and T. Hu, "Safe primes construction algorithm based on laime primes judgment theorem," *Computer Engineering and Applications.*, vol. 52, no. 13, pp. 152–156, 2016.

Biography

Hexiong Chen, born in 1984, received the master degree from Kunming University of Science and Technology in June 2011, and now working at Yunnan Power Grid Company Limited. His research interests include Network Technology, Network Security, Network Operation and Maintenance.

Jiaping Wu, born in 1998. He is currently a M.S. candidate in University of Electronic Science and Technology of China. His research interests include Internet of Things Security and Blockchain Technology.

Wei Guo, born in 1986, received the bachelor degree and now working at Yunnan Power Grid Company Limited. His research interests include Network and Network Security Maintenance and Management.

Feilu Hang, born in 1984, received the master degree from Yunnan University in June 2014, and now working at Yunnan Power Grid Company Limited. His research interests include Network Security Attack and Defense Technology.

Zhenyu Luo, born in 1985, received the bachelor's degree and now working at Yunnan Power Grid Company Limited. His research interests include Network Technology, Network Security, Network Operation and Maintenance.

Yilin Wang, born in 1998. She is currently a M.S. candidate in University of Electronic Science and Technology of China. Her research interests include Internet of Things Security and Blockchain Technology.