Research Article

A Secure Anonymous Identity-Based Virtual-Space Agreement Method for Crowds-Based Anonymous Communicate Scheme

Kai Lin,1,2 Kaiyu Wang,1 Jin Shang,1 and Qindong Sun1

1The School of Cyber Security, Xi’an Jiaotong University, Xi’an, China
2Sichuan Digital Economy Industry Development Research Institute, Chengdu, China

Correspondence should be addressed to Qindong Sun; qdongsun@xjtu.edu.cn

Received 31 May 2023; Revised 21 September 2023; Accepted 30 November 2023; Published 18 December 2023

Abstract

Anonymous data exchange is in great demand in many situations, especially in remote control systems, in which a stable, secure, and secret data channel must be established between the controlling and controlled parties to distribute control commands and return data. In the previous work, we built a two-level Virtual-Space anonymous communication scheme based on the Crowds System for performing secret data exchange in remote control systems. However, as an essential part of security and anonymity, participating nodes’ identity declaration and session key agreement phases were not well designed. In this paper, we redesign the identity agreement and declaration process and design an identity-based Virtual-Space agreement method using the extended Chebyshev Chaotic Maps. In this approach, we transform the identity declaration process into a multilevel Virtual-Space agreement problem, where a series of security-progressive Virtual-Space addresses are negotiated between the controller and the controlled nodes. The protocol can handle the case where there are multiple controllers in the system, and the negotiated Virtual-Space depends on the identity of the controller and the controlled node, so different controllers do not affect each other. The designed protocol is verified on Freenet, and we conclude this paper with a detailed security analysis of the method to prove that the method satisfies forward security.

1. Introduction

The development of the Internet has dramatically reduced the cost of data exchange, making the exchange of data over ultra-long distances an extremely low-cost affair. Messaging using the Internet requires that the location information of the sender and receiver of the message be encoded in the transmitted message in some way so that the message can arrive as expected. In the development of the Internet, data transmission has gradually changed from the initial plaintext transmission to encrypted transmission, and encrypted transmission techniques, including Transport Layer Security, have been adopted to protect the contents of the communication content.

However, the anonymity of the communicating parties’ identity information is as important as the security of the communication content in some scenarios, such as network attacks and remote control, in which the attacker needs to protect his identity from being discovered and, therefore, needs to use covert methods to communicate with the attacked object. In addition, some users also need to access some network services covertly without revealing their identities. There are various methods to hide the identity information of the communicating parties from the intermediate nodes of the network, such as using proxies or virtual private networks. To provide stronger anonymity protection, scholars have designed specialized anonymous communication systems, such as Tor [1], I2P [1] based on the mix [2] mechanism, and Freenet [3] based on the Crowds [4] mechanism, to achieve anonymous Internet-based communication, which prevents intermediate processing nodes of messages or network eavesdroppers from analyzing the identities of the communicating parties.

Since the anonymity provided by these anonymous communication systems fits the requirement of anonymity of communication for remote control systems, we try to use these anonymous communication systems to build an anonymous control system for remote control system command...
distribution and data return. In our previous work [5], we designed an anonymous control scheme based on the Crowds system.

In the scheme designed by Sun et al. [5], the controller’s identity is the most critical information, so its protection is of top priority. A two-level Virtual-Space-based anonymous control scheme is designed for the exchange of data between the controller and the controlled nodes by Sun et al. [5], where the first level Virtual-Space, called identity space, is used for identity announcement and identity information exchange, and the second level Virtual-Space, called message space, is used for message transmission.

The identity space is computed periodically based on the common knowledge shared between the controller and the controlled node, and the address of the message space is encrypted by each node and published into the computed identity space. The use of Virtual-Space for communication makes any node in the remote control system unaware of the real identity of the controller; they can verify that a message is from a legitimate controller but cannot obtain any relevant information about the controller. The main problem with this scheme is that the identity space is generated in an overly simple way that anyone who gets the common knowledge as long as the algorithm can calculate the address of the identity space and obtain the number of online nodes from the identity space. In addition, the scheme lacks flexible control authority control, in which only one controller identified by a public key precoded in all controlled nodes is allowed in the system. In order to remedy the problems mentioned above of the scheme by Sun et al. [5], this paper designs an identity agreement and declaration protocol for Virtual-Spaces and uses a digital certificate-based authentication scheme to achieve flexible controller permission control. Considering that the original scheme can be applied to the Internet of Things (IoT) environment, where most smart terminals are resource-constrained devices, the data agreement protocol in this paper is based on the Chebyshev Chaotic Map (CCM), which can minimize the computational overhead while achieving high security [6]. Based on the discrete logarithm problem and the Diffie–Hellman problem of Chebyshev chaotic mapping, the proposed protocol can negotiate confidential data in an insecure communication channel for both the controller and the controlled node, which is known only to the communicating parties. A thorough theoretical analysis of the protocol demonstrates that the design protocol in this paper is secure and can protect the controller’s identity information in the presence of active attackers in the network. The protocol is proved to be practically usable by constructing experiments on Freenet. The main contributions of this paper are as follows:

(i) A multilevel Virtual-Space-based identity and Virtual-Space agreement mechanism is designed to improve our previously designed Virtual-Space-based anonymous control scheme, providing stronger security for this anonymous control scheme;

(ii) Considering the computational resource limitation of the nodes in the remote control scheme, Chebyshev chaotic mapping is used as a mathematical tool for negotiating secret data, thus reducing the computational resource consumption in the agreement process.

(iii) A detailed security analysis of the proposed identity agreement scheme is provided, demonstrating that the scheme meets the security requirements of the anonymous remote control system.

The remainder of the paper is organized as follows: Section 2 introduces some existing works related to anonymous key agreement protocols and anonymous authentication protocols associated with CCM; Section 3 presents some background knowledge of the paper, including an introduction to CCM and an introduction to Virtual-Spaces; Section 4 gives the detailed design of the protocol; Section 5 discusses the security of the protocol in detail; Section 6 implements the protocol prototype in Freenet and analyzes the operational efficiency of the prototype; and finally, Section 7 concludes the paper.

2. Related Work

A chaotic system is defined as a system that is highly sensitive to initial conditions [7], whose sensitive dependence on initial conditions and similarity to random behavior are essentially the same as required by several cryptographic primitives [8]. CCM, also known as Chebyshev polynomials, originating from the cosine and sine functions of multiplicative angles, have a simple implementation, low computation cost, and good chaotic behavior [9]. Since Chebyshev polynomials have two computationally hard problems, chaotic map-based discrete logarithm problem (CMBDLP) and chaotic map-based diffie-hellman problem (CMBDHP), Chebyshev polynomials can be used for designing key agreement protocols or authentication protocols with high security while minimizing computational cost [6]. Researchers have been focusing on developing lightweight cryptographic protocols due to the increasing use of wireless sensor network (WSN) and IoT technology. This is because most of the devices used in these networks are resource-constrained, and using traditional cryptographic algorithms could overload them, which may lead to additional risks [10]. In this context, the Chebyshev polynomial has gained much attention due to its lightweight nature and has been used to design lightweight cryptographic protocols such as key agreement protocols and authentication protocols.

Lee et al. [11] proposed an efficient anonymity key agreement protocol based on Chebyshev polynomials, which ensures user anonymity while also achieving mutual authentication between the server and the user. Most importantly, their protocol overcomes the weaknesses of previous authentication protocols designed by Xiao et al. [12]. Abbasinezhad-Mood and Nikooghadam [13] designed an efficient anonymous password-authenticated key exchange protocol using extended Chebyshev polynomials, which can be applied in smart grid environments with limited computational resources. Cui et al. [14] proposed a full session key agreement scheme based on
Chebyshev polynomials for the vehicular ad-hoc network, which avoids the modular multiplication index or scalar multiplication on the elliptic curve by using Chebyshev polynomials. Wang et al. [15] developed a secure authentication key agreement scheme for smart grid environments.

Guo et al. [16] proposed a three-factor authentication scheme based on Chebyshev polynomials for session initiation protocol, which can provide secure mutual authentication between a user and the remote server. Lee et al. [17] proposed an efficient single-sign-on authentication mechanism using extended Chebyshev polynomials, which can be used to authenticate users in a distributed environment, reduces the amount of data transfer and computing resources required during the authentication process while ensuring security. Meshram et al. [18] proposed an efficient online/offline ID-based short signature procedure using extended Chebyshev polynomials that have very minimum operation. Zhang et al. [6] adopted a square in every process, which is very suitable for resource-constrained environment such as WSN. Zhang et al. [6] extended CCM. Chebyshev polynomial map (PKC) infrastructures that may need to be modified. Meshram et al. [19] proposed an efficient remote authentication scheme by combining convolution-CCM with biometric techniques. Similarly, Nyangaresi [20] developed a message authentication protocol for WSN environments by combining biometrics with extended CCM.

Thanks to the lightweight nature of Chebyshev polynomials, most of these schemes or protocols based on Chebyshev chaotic achieve better computational performance than similar schemes designed using other cryptographic algorithms, such as RSA. These protocols require fewer computational or communication resources than others. However, with the growing use of Chebyshev chaotic in designing lightweight asymmetric cryptographic schemes, there are other related issues that require attention, such as cryptographic models designed for existing public key cryptography (PKC) infrastructures that may need to be modified. Meshram et al. [21] proposed a robust and secure identity-based encryption transformation model for PKC using CCM.

However, public key crypto-systems constructed using Chebyshev polynomials working on real number fields are noted to be insecure [7]. In order to implement more secure and practical public key algorithms, Chebyshev polynomials are extended from the real domain to finite fields and finite rings [22]. Liao et al. [23] then analyzed Chebyshev polynomials working on the finite field $Z_N$ and proved that Chebyshev polynomials working on the integer ring $Z_N$ are not secure in the sense of cryptography when $N$ is not chosen properly [23, 24]. They also point out that Chebyshev polynomials working on the integer ring $Z_N$ can achieve sufficient security when $N$ is carefully chosen and that $N$ is several strong primes’ products with small powers. In other words, $N = \prod_{l=1}^{k} p_i^{x_i}$, in which $p_i$ is a prime with $p - 1 = 2q_1$ and $p + 1 = 2q_2$ where $q_1$ and $q_2$ are also primes. To reduce the time complexity of computing Chebyshev polynomials, Zhang et al. [6] proposed a binary exponentiation algorithm based on square matrices.

3. Background

3.1. Extended Chebyshev Chaos Map. CCM (also known as Chebyshev polynomials, which are used in the rest of the paper) can be defined as follows [6, 9]:

$$T_n(x) = \cos(n \times \arccos(x)),$$

where $n \in \mathbb{N}$, and $x \in [-1, 1]$. Alternatively, Chebyshev polynomials can also be defined recursively as follows:

$$T_n(x) = \begin{cases} 1 & n = 0 \\ x & n = 1 \\ 2xT_{n-1}(x) - T_{n-2}(x) & n \geq 2 \end{cases}$$

Equation (3) is an example of the Chebyshev polynomials when $n$ is taken from 2 to 5 [11].

$$T_n(x) = \begin{cases} 2x^2 - 1 & n = 2 \\ 4x^3 - 3x & n = 3 \\ 8x^4 - 8x^2 + 1 & n = 4 \\ 16x^5 - 20x^3 + 5x & n = 5 \end{cases}$$

Zhang [25] enhances the Chebyshev polynomials and extends the range of $x$ to $(-\infty, +\infty)$. When $x \in Z_N$, extended Chebyshev polynomials can be defined recursively as follows [6, 24]:

$$T_n(x) = \begin{cases} 1 & n = 0 \\ x & n = 1 \\ 2xT_{n-1}(x) - T_{n-2}(x) \mod N & n \geq 2 \end{cases}$$

3.1.1. Properties of Chebyshev Polynomials. Chebyshev polynomials have two significant properties, which are the chaotic property and the semigroup property.

(1) The Chaotic Property. Chebyshev polynomial map

$$T_n: [-1:1] \rightarrow [-1,1],$$

with degree $n>1$ is a chaotic map with its invariant density $f^\ast(x) = \frac{1}{\pi\sqrt{1-x^2}}$, for positive Lyapunov exponent $\lambda = \ln n > 0$ [11, 19].

(2) The Semigroup Property. The semigroup property of Chebyshev polynomials can be described as follows:

$$T_r(T_s(x)) = T_{r+s}(x).$$

where $r$ and $s$ are both positive integers and $x \in [-1,1]$ [8, 17, 24]. This property allows compounding Chebyshev polynomials to obtain a new polynomial [15].

3.1.2. Computational Problems of Chebyshev Polynomials. Chebyshev polynomials present two challenging computational
problems that make them suitable for designing cryptographic protocols like key agreement and remote authentication. These issues are referred to as CMBDLP and CMBDHP [6, 13, 15, 19].

(1) Chaotic Map-Based Discrete Logarithm Problem. Given \( x \) and \( y \), it is not feasible to compute or find \( n \) in polynomial time that makes \( T_n(x) = y \).

(2) Chaotic Map-Based Diffie–Hellman Problem. Given \( T_s(x) \) and \( T_u(y) \), it is not feasible to compute or find \( T_{vu}(x) \) in polynomial time.

3.2. Virtual-Space. Virtual-Space is a concept that we have proposed in our previous work [5]. Specifically, [5] designed an anonymous communication scheme based on the Crowds system for a remote control scenario, in which the data is stored in a set of nodes that can be indexed by a static key Key. Logically, each Key corresponds to a space in the network, and a large number of files can be stored in each space. In this scheme, the operation of sending data can be considered as depositing data in the network space identified by a given Key, while requesting data can be considered as retrieving specific data from the space identified by a given Key. But in reality, there is no space in the network, and that is the reason for the Virtual.

Actually, each Key identifies a location value, and according to the Key a set of nodes with location values similar to the location value identified by the Key can be found, and these nodes will store the data corresponding to the Key. Since the existence of Key well abstracts the specific details of data storage, the users of the scheme need not care about the specific data storage location but only need to know that a Key can be used to store a set of data, so a Key can be considered a Virtual-Space.

4. The Identity Agreement and Declaration Method

A traditional remote control system can be seen as a star-shaped network model, with a central controller node and many controlled nodes, in which controlled nodes are only responsible to the central controller node. In contrast, the method described next describes a distributed control model in which multiple controllers are allowed to perform control operations on all controlled nodes simultaneously, and these controllers can use the same or different nodes to issue control commands. There are no controller nodes in the model but only controlled nodes and controllers. The identity declaration process of the controlled nodes is treated as a process of negotiating Virtual-Spaces with the controller, in which each controlled node independently negotiates a series of Virtual-Space addresses with increasing security with the controller, of which the most secure Virtual-Space will be used to exchange the private data. Table 1 lists some of the notations used in the remainder of this section, and Figure 1 depicts the overall flow of the scheme, the details of which are described later in this section.


The model uses digital certificates to manage the hierarchical structure of controllers. Consider a controlled group \( \mathbb{G} \) with \( m \) controlled nodes; the set of controlled nodes \( \mathbb{G} \) contains can be defined as \( \mathbb{G} = \{ \text{Node}_i | 0 \leq i < m \} \), and the user \( U_r \) who created the controlled group is called the root controller. \( U_r \) first creates a root CA, internalizes the CA’s public key \( \text{CA}_{pub} \) in all \( \text{Node}_i \in \mathbb{G} \), and the CA’s private key \( \text{CA}_{priv} \) is stored securely by \( U_r \) offline. \( U_r \) designates the actual controller of

<table>
<thead>
<tr>
<th>Notations</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U_r )</td>
<td>The root controller</td>
</tr>
<tr>
<td>( U_n )</td>
<td>The nth actual controller</td>
</tr>
<tr>
<td>( \mathbb{C} )</td>
<td>The set of all actual controllers</td>
</tr>
<tr>
<td>( \text{CA}<em>{priv}, \text{CA}</em>{pub} )</td>
<td>Digital certificate issued by ( U_r ) for ( U_n ), its private key and public key</td>
</tr>
<tr>
<td>( D_n, D_n^{priv}, D_n^{pub} )</td>
<td>G represents a controlled group, and ( \mathbb{N} ) represents the set of all controlled nodes ( \mathbb{G} ) contains</td>
</tr>
<tr>
<td>( \mathbb{G}, \mathbb{N} )</td>
<td>The ith controlled node</td>
</tr>
<tr>
<td>( \text{ID}_i, I )</td>
<td>Identifier of ( \text{Node}_i ), and ( I ) is the set of all ( \text{ID}_i )s of ( \mathbb{N} )</td>
</tr>
<tr>
<td>( \text{HID}_n )</td>
<td>Hash of ( \text{ID}_n ) for ( U_n )</td>
</tr>
<tr>
<td>( T_n(x) )</td>
<td>Chebyshev polynomials function</td>
</tr>
<tr>
<td>( C_j )</td>
<td>The jth identity declaration cycle</td>
</tr>
<tr>
<td>( K_n, K_n )</td>
<td>Public knowledge for ( C_j ). In particular, the ( K_n ) obtained by ( U_n ) is noted as ( K_n ).</td>
</tr>
<tr>
<td>( S_{C_j} )</td>
<td>The first level of identity space</td>
</tr>
<tr>
<td>( S_{C_j}, S_{C_j} )</td>
<td>The second level of identity space between ( U_n ) and ( \text{Node}_i ) of ( C_j )</td>
</tr>
<tr>
<td>( \text{SM}<em>{C_j}, \text{SM}</em>{C_j} )</td>
<td>The message space between ( U_n ) and ( \text{Node}_i ) of ( C_j )</td>
</tr>
<tr>
<td>( \text{IM}<em>{C_j}, \text{IM}</em>{C_j} )</td>
<td>The identity file of ( U_n ) in ( C_j ) and the identity message of ( U_n ) in ( C_j )</td>
</tr>
<tr>
<td>( E_k(x) )</td>
<td>Cipher obtained by encrypting ( x ) with key ( k )</td>
</tr>
<tr>
<td>( \oplus, |, \text{Hash}(x) )</td>
<td>XOR, concatenation, and one-way hash function, respectively</td>
</tr>
</tbody>
</table>
G without actually participating in it. For the \( j \)th identity declaration cycle \( C_j \), \( U_r \) create a set of controllers \( C \) with \( N \) actual controllers and issues a digital certificate \( D_n \) using \( CA_{priv} \) for each \( U_n \in C \), and the validity period \( c''_n \) of \( D_n \) satisfies \( c''_n < c''_{n+1} \).

For any \( U_n \), he can arbitrarily pick \( Node^* \in \mathbb{N} \) to give control commands to \( G \). Before \( U_n \) can start the control operation for every identity declaration cycle \( C_j \), \( U_n \) must complete the identity declaration and then negotiate a data space \( SM_{i,n} \) shared with each \( Node_i \in \mathbb{N} \). For any \( i \neq i_j \) or \( n \neq n_j, SM_{i,n} \neq SM_{i,n_j} \), thus, for any \( C_j \), there will be up to \( \mathcal{m} \times N \) Virtual-Space \( SM_i \). The identity declaration and the agreement of \( SM_i \) will be described specifically in the next section.

4.2. Proposed Method. When \( U_r \) creates \( N, U_r \) assigns a random secret ID \( ID_i \) to each \( Node_i \in \mathbb{N} \). For any \( i \neq i_j \), \( ID_i = ID_j \). The set of all \( ID_i \) of \( \mathbb{N} \) is denoted as I. \( ID_i \) is only shared between the \( U_r \) and the \( Node_i \) and is not available to anyone else, including \( U_n \). After \( U_r \) issues the digital certificate \( D_n \) for \( U_n \), \( U_r \) will use \( D_n \) to calculate a \( HID_i \) for each \( Node_i \) using Equation (7), and then give \( D_n \) to \( U_n \) together with all the \( HID_i \). Due to the use of \( D_n \) in the calculation process, \( HID_i \) has the same validity period as \( D_n \).

\[
HID_i = \text{Hash}(ID_i || \text{Hash}(D_n)). \tag{7}
\]

The process of Virtual-Space agreement can be divided into three steps: public knowledge acquisition, identity declaration, and secret space agreement. Each process will output a Virtual-Space address, increasing the output Virtual-Space security. The Virtual-Space address produced by the secret space agreement step is shared only between \( U_n \) and \( Node_i \), to transmit confidential data.

4.2.1. Public Knowledge Acquisition. In the first step of Virtual-Space agreement of \( C_j \), \( U_n \), and \( Node_i \), need to acquire the same public knowledge \( K_j \) and use \( K_j \) as a seed to generate a shared Virtual-Space, that is, \( S_{i,n} \), through a shared function \( f(x) \). \( S_{i,n} \) is calculated as shown in Equation (8)

\[
S_{i,n} = f(\text{Hash}(K_j)). \tag{8}
\]

\( K_j \) should satisfy the following two properties:

(i) For any \( C_j (j < j) \), it is almost impossible for different participants to obtain the same \( K_j \).

(ii) For any \( C_j (j \geq j) \), different participants can always obtain the same \( K_j \).

In other words, \( K_j \) should be unpredictable, and there is no efficient way to obtain \( K_j \) in the future. For these reasons, \( K_j \) should be obtained from an unpredictable public data source, such as a publicly published unpredictable data source on the Internet.

In addition, the selected public data source should have sufficient access traffic that any Internet participant should have a very high incentive to access the data source and obtain \( K_j \) to ensure that \( U_n \) and \( Node_i \) will not reveal their identities because of requesting \( K_j \). An example data source for \( K_j \) is the opening stock price of Google Inc. Since the stock market is tough to predict, \( K_j \) obtained by querying the opening stock price can be approximated as unpredictable.
Due to the large size of the stock market, there is sufficient incentive for any nodes to obtain Google’s stock price at any point in time.

4.2.2. Identity Declaration. The $SI^j_i$ is insecure because both $K_n$ and $f(x)$ are public, which means that anyone can calculate the same $SI^j_i$ based on $K_n$ and $f(x)$ and then can read or write the data in the $SI^j_i$. Therefore, $SI^j_i$ cannot be used to transfer confidential data, and a more secure Virtual-Space needs to be further negotiated between $U_n$ and $Node_i$. After calculating the $SI^j_i$, $U_n$ selects a random number $p$ and a large prime number $m$ and then calculates its identity data $IM^j_i$ using Equation (9)

$$IM^j_i = T_p(\text{Hash}(K_n)) \mod m,$$  

where $T_p(x)$ is Chebyshev polynomials function, and $K_n$ is the public knowledge acquired by $U_n$. Subsequently, $U_n$ uses $D^{pri}_n$ to sign the $IM^j_i$ with $m$ to obtain the signature file $DS_n$, and get the identity file $IF^j_i$ by Equation (10)

$$IF^j_n = IM^j_n \| m \| DS_n \| D_n.$$  

$U_n$ then uploads the $IF^j_i$ to the $SI^j_i$ in order to announce itself to $\mathbb{N}$. Subsequently, $U_n$ calculates secondary identity space for every $Node_i \in \mathbb{N}$ using Equation (11)

$$SI^j_{in} = \{ f(HID^j_p) \| D_n \}$$

$$\oplus \text{Hash}(IM^j_n \| HI^j_n) \}.$$  

$Node_i$ periodically retrieves the $SI^j_i$ until it fetches the identity file $IF^j_i$ uploaded by $U_n$. Once $Node_i$ gets the $IF^j_i$, it verifies the legitimacy of the $D_n$ using the built-in CA public key $CA_{pub}$ and then uses the $D_n$ to verify the legitimacy of the $DS_n$. When all checks pass, $Node_i$ calculates the $SI^j_{in}$ using Equation (11). $SI^j_{in}$ is more secure than $SI^j_i$ since $SI^j_{in}$ is calculated from the confidential attribute $HID$ of $Node_i$, which is known only to $U_{root}$, $U_n$, and $Node_i$. However, since $U_n$ and $Node_i$ need a Virtual-Space shared only between them, they need to negotiate further to get more secure Virtual-Space.

4.2.3. Secret Space Agreement. After $Node_i$ computes the $SI^j_{in}$, it picks a random number $q$ and computes $q_i$ through Equation (12)

$$q_i = T_q(\text{Hash}(K_n)) \mod m.$$  

$Node_i$ then encrypts $q_i$ using $D_n$ to obtain $Eq_i = E(D_n, q_i || SALT)$, where $SALT$ is a large, random string that $U_n$ can easily distinguish from $q_i$ to protect $Eq_i$ from chosen-plaintext attack. $Node_i$ uploads $Eq_i$ to $SI^j_{in}$ and waits for $U_n$ to retrieve $Eq_i$.

Finally, $Node_i$ and $U_n$ can calculate the final Virtual-Space $SM^j_{in}$ by $IM^j_i$ and $q_i$, which is shown as below:

$$SM^j_{in} = T_q[T_p(\text{Hash}(K_n))] \mod m$$

$$\equiv T_p[T_q(\text{Hash}(K_n))] \mod m$$

$$\equiv T_p[HID_n] \mod m.$$

$SM^j_{in}$ is secure enough to be used to transmit secret data, as $SM^j_{in}$ is shared only between $U_n$ and $Node_i$. Since both $U_n$ and $Node_i$ have permission to write data to $SM^j_{in}$, $U_n$ and $Node_i$ can subsequently communicate using an asymmetric key-based Virtual-Space and publish the data read address of the Virtual-Space in $SM^j_{in}$ without worrying about other nodes or users other than $U_n$ or $Node_i$ getting the data.

5. Security Discussion

This section discusses the security of the proposed method from a theoretical point of view. First, the threat model is given, followed by an analysis of the security requirements in the threat model to show that the protocol can satisfy these security requirements.

5.1. Threat Model. The threat model in this paper is based on the widely accepted Canetti and Krawczyk (CK) threat model [26], also known as the CK threat model. In this threat model, a probabilistic polynomial-time attacker has full control over the communication link and can eavesdrop, alter, drop, delay, and inject into the transmitting information. The attacker can also control the scheduling of all protocol events, including the initiation of protocols and message delivery. In addition, an attacker can compromise one of the protocol participants $P$ and thus obtain all the local states stored in the $P$ about the session. In this paper, there exist two types of participants $P$, namely $Node_i$ and $U_n$. According to the summary of the CK threat model by Abbasinezhad-Mood and Nikooghadam [13] and Wang et al. [15], there are seven operations that attacker $\mathcal{A}$ can carry out, which include:

1. $(1)$ Execute$(Node_i, U_n)$. This operation represents passive eavesdropping attacks and returns information exchanged between $Node_i$ and $U_n$.
2. $(2)$ Send$(P, m)$. This operation represents active attacks that $\mathcal{A}$ can send any messages to protocol participants $P$ and receive the response from $P$.
3. $(3)$ ESRreveal$(P)$. This operation allows $\mathcal{A}$ to obtain the ephemeral key of the specified participant $P$ and the internal state of the specified session stored in $P$.
4. $(4)$ SKReveal$(P)$. This operation allows $\mathcal{A}$ to obtain the final session key of $P$.
5. $(5)$ Corrupt$(P)$. This operation allows $\mathcal{A}$ to obtain all the information about $P$, including $P$’s long-term secret.
6. $(6)$ Expire$(P)$. This operation allows $P$ to completely delete all information related to a specified session, including the session key, and the deleted information can no longer be accessed in any way.
7. $(7)$ Test$(P)$. This operation can be used to test the semantic security of the session key. Upon receiving
a Test(P) query, P toss a unbiased coin $b, b \sim \{0,1\}$. if $b = 1$, the actual session key is returned; otherwise, a random value with the same length is returned.

Combined with the application scenario of the scheme in this paper, we assume that attack goals of $\mathcal{A}$ are as follows:

(i) To obtain the identities of Nodei. If $\mathcal{A}$ compromises a node Nodei, $\mathcal{A}$ will try to use Nodei to discover the identity of other nodes Nodej, where $i \neq j$.

(ii) To obtain the identities of $U_n$ or $U_r$ by various methods. $\mathcal{A}$ can eavesdrop on messages traveling through the network, compromise a node, and infer the identities of $U_n$ or $U_r$ from the session-local state stored in the node.

(iii) To obtain the $SM_{i,n}$ address negotiated between $N$ and Un. This attack only considers the case where $\mathcal{A}$ has not compromised Nodei. When $\mathcal{A}$ compromises Nodei, it can read $SM_{i,n}$ from Nodei’s memory, and then $\mathcal{A}$’s attack goal is to obtain the message space $SM_{i,n}$ negotiated by other nodes Nodek($k \neq i$) and $U_n$.

(iv) Disguise as $U_n$ and release data to Nodei. This attack assumes that $\mathcal{A}$ has all the protocol’s key algorithms but cannot obtain or use the identity information of any Nodei. $\mathcal{A}$ will create an identity message in this case and attempt to protocol with $U_n$ or $U_r$.

Before further analyzing the scheme’s security, we give the assumptions for the analysis. First, we assume that $U_n$ and $U_r$ are honest but curious. They do not wish to interrupt other controllers’ sessions but are interested in the messages transmitted to Nodei by other controllers. Second, we assume that $U_n$ and $U_r$ are secure and that the nodes they use are not compromised by $\mathcal{A}$, which is a reasonable assumption because $U_n$ and $U_r$, as the controllers of the remote control system, will always be in a secure place and use a secure node to distribute messages to the controlled.

5.2. Session. The further analysis uses Session to represent a temporary data exchange period between the controller $U_n$ and the node Nodei that uses $SM_{i,n}$ to exchange data. A session can be defined as follows:

**Definition 1 (Session).** A session $S_{i,n,j}$ is a temporary information interchange between controller $U_n$ and Nodei during identity declaration cycle $C_{ij}$ which can be represented as a ternary:

\[ S_{i,n,j} = <C_{ij}, U_n, Nodei>. \]  \hspace{1cm} (14)

Two sessions $S = S_{i,j,n}$ and $S' = S_{i',j',n'}$ are considered consistent iff $i = i'$, $j = j'$, and $n = n'$.

5.3. Anonymity. Before further security analysis, it is necessary to explain a key security concept, the anonymity. There is a slight difference in the meaning of anonymity in anonymous communication systems and anonymous authentication protocols. In an anonymous communication system, anonymity is the state of being not identifiable within a set of the anonymity set [27], and related concepts include unobservability and pseudonymity. Anonymity in anonymous communication systems can be described in a variety of ways, such as the degree of anonymity [4, 28] and the description from the attackers’ perspective [29]. Meanwhile, there exist various ways to measure the anonymity of an anonymous communication system, such as methods based on information theory [28, 30–34]. Overall, anonymized communication systems primarily consider observers who are not directly involved in the communication, and anonymity in an anonymized communication system describes the ability of an observer to obtain information about the identity of the users who are communicating with each other. The higher the anonymity is, the more difficult for an observer to associate network traffic with specific users.

In an anonymous authentication protocol or an anonymous key exchange protocol, the meaning of anonymity is much simpler than that in anonymous communication systems. Unlike the anonymity of anonymous communication networks, which has been exhaustively formalized and extensively discussed, the anonymity of such protocols lacks a formal definition. However, from the security analysis of much anonymous authentication protocols/anonymous key exchange protocols, in most instances, the anonymity of such protocols means that the participants’ real identities are not involved in authentication and data transmissions [35], the probability that an attacker can determine the true identities of the protocol participants by executing the protocol [36] or observing the packets generated during the execution of the protocol is negligible [37]. In some protocols, pseudonyms are utilized to safeguard the anonymity of users, such as [37, 38]. Cao and Wei [36] defined user anonymity as shown in Equation (14). The definition of anonymity for this paper can be obtained by modifying the definition method of Cao and Wei [36] for a two-factor protocol. The user anonymity in our protocol can be defined as follows:

**Definition 2 (Anonymity).** The protocol achieves user anonymity if the following equation holds:

\[ ADV_{\text{anon}} = Pr[Succ(\mathcal{A})] = Pr[c' = c] \leq \epsilon, \]  \hspace{1cm} (15)

where $c'$ is the identity of the protocol participant guessed by probabilistic polynomial-time adversary $\mathcal{A}$ by up to $q$ times active or passive attacks, $c$ is the true identity of the protocol participant, and $\epsilon$ is a sufficiently small negligible constant.

To summarize, in anonymous authentication or key exchange protocols, anonymity means that none of the participants in the session can gather information about the true identity of other participants $P$. In such protocols, anonymity only considers the information in the packets generated during the protocol interaction. It does not consider the case...
where an attacker $A$ uses information outside the protocol, such as the IP address of the transmission process or data traffic characteristics, to obtain the true identity of $P$, which is considered by an anonymous communication system.

The work in this paper is to provide a set of Virtual-Space agreement methods [5] for a previously designed anonymous communication system, which incorporates authentication of the participating communication nodes. Thus, the essence of the work in this paper is an anonymous key agreement and anonymous authentication protocol. With the discussion of anonymity above, analyzing anonymity in the subsequent security analysis does not consider the case where an attacker $A$ uses information outside the protocol to infer the identity of a protocol participant $P$. In the subsequent analysis, we assume that $P$ uses the anonymous communication system commonly and securely, and communication characteristics such as $P$’s IP address and traffic characteristics are invisible to $A$. $A$ can only obtain the true identity of $P$ by interacting with the protocol, interpreting its contents, and implementing the attack methods defined in Section 5.1.

5.4. Security Analysis. The next part of this section analyzes the security of the proposed method in the following aspects and discusses whether it can cope with the previously introduced threat model.

5.4.1. Forward Security. Günther [39] first introduced the concept of forward security in 1989 that even if the secret key $K$ of a key authentication center is known by accident, the confidentiality of past messages in sessions constructed from $K$ can not be compromised. Colin and Kai [40] classify forward security, namely Absolute Forward Security, Delayed Forward Security, and Null Forward Security, based on the period of data that an adversary can obtain after the confidentiality of $K$ was compromised. Ran Canetti and Hugo Krawczyk [26], in the discussion of the nature of perfect forward secrecy (PFS), argue that a key-exchange protocol that has a mechanism for expiring a session (see $Expire(P)$ in Section 5.1) can get proof automatically that this protocol guarantees PFS if this protocol can be proved by the definition of SK-secure.

Ran Canetti and Hugo Krawczyk define SK-secure as well as SK-secure without PFS as follows [26]:

**Definition 3 (SK-Secure).** A key-exchange protocol is called **SK-secure** if the following properties hold for any adversary $A$ in the Unauthenticated Links Model:

1. If two protocol participants completed the protocol under the same session, then they will both output the same key.
2. The probability that $A$ guesses correctly the bit $b$, that is, $b' = b$, is no more than $1/2$ plus a negligible fraction in the security parameter.

**Definition 4.** A key-exchange protocol is called **SK-secure without PFS** if the key-exchange protocol is SK-secure but is not allowed to expire keys.

It is clear that both parties involved in the protocol outputting the same Virtual-Space, $T_{pq}(Hash(K_n)) \mod m$. As the second point of Definition 3, due to the chaotic property of CCM, it can be considered that it is equally likely that each bit of the final output is taken to be either 0 or 1, that is, $Pr\{b = 0\} = Pr\{b = 1\} = 1/2$, thus the attacker’s probability of guesses correctly for each bit is 1/2. As introduced in Section 2, CCM has been widely used in the design of key agreement/authentication protocols, and the security of CCM is given a detailed analysis in the study of Liao et al. [23]; therefore, it can be considered secure to apply CCM. Thus, the protocol proposed in this paper satisfies the definition of SK-Secure, and the discussion of PFS of the protocol can be translated into the discussion of whether the protocol supports session expiration.

**Theorem 1.** The protocol in this paper allows for session expiration. Once expired, the session key $SM_{i,n}$ of the expired session cannot be recovered in any way.

**Proof.** As the definition of the Session in Definition 1, it is easy to see that different identity declaration cycles $C_i$ will generate different sessions, and session $S_j$ will expire when $C_j$ ends. Thus, the proof focuses on showing that the session key $SM_{i,n}$ is unrecoverable, even if the long-term secret of the control group $\mathcal{G}$ leakage.

The protocol has three types of long-term secret data: $CA_{priv}, D_n^{priv}$, and the list of all node identities, $\mathcal{L}$. Consider the worst-case scenario, where the attacker $A$ can access all three types of secret data. In this scenario, $A$ has access to all the data before the secret space agreement step, including $IM_{i,n}, IP_{n}, SL_{i,n}$, and $q_i$. To recover the session key $SM_{i,n}$, all $A$ needs to do is complete the computation of Equation (13). However, due to CMBDLP, it is difficult for $A$ to compute in polynomial time through $IM_{i,n}$ and $q_i$ to obtain the random numbers $p$ and $q$ selected by $U_n$ and $Node_i$, respectively, which are never transmitted on the network. Also, due to CMBDHP, it is not feasible for $A$ to get the value of $SM_{i,n}$ through $IM_{i,n}$ and $q_i$. Thus, even in the worst case, $A$ cannot recover an expired session key using the secret information obtained. The other case, where $A$ gets only one or two types of secret data, is included in the worst scenario and can easily get that $A$ cannot recover the expired session key $SM_{i,n}$.

From the proof of Theorem 1, it is easy to conclude that for an active attacker $A$ who can carry out operations $Execute(Node_i, U_n)$, $Send(P, m)$, $Expire(P)$, and $Test(P)$ defined on Section 5.1, $SM_{i,n}$ is secure since $A$ cannot obtain the random numbers $p$ and $q$.

**Corollary 1.** For any attacker $A$, the session key $SM_{i,n}$ is secure when $A$ cannot obtain the random numbers $p$ and $q$.

In conclusion, the protocol satisfies Definition 3 and has a mechanism for expired sessions, where expired session keys cannot be recovered, so the protocol can get proof automatically that the protocol guarantees PFS through the discussion of Ran Canetti and Hugo Krawczyk.
Corollary 2. The protocol in this paper satisfies SK-secure with PFS.

5.4.2. Anonymity of Nodei. Based on the discussion of anonymity in Section 5.3, the anonymity of Nodei in this protocol mainly refers to the security of Nodei’s identity, that is, whether Un or a polynomial attacker $\mathcal{A}$ can obtain Nodei’s identity $ID_i$ during the protocol. Since $ID_i$ is the confidential information allocated by $U$, and shared only between Nodei and Un, which affects the result of the protocol, we can consider that the protocol satisfies the anonymity of Nodei if no one except Nodei and Un can obtain $ID_i$ in any way during the protocol.

Theorem 2. For anyone, except Nodei and Un, the probability that $ID_i$ satisfies $ID_i = ID_i'$ is no more than $e$, which is negligible.

Proof. For any $i \neq i'$, $ID_i \neq ID_i'$. The information available to $U_n$ related to the identity of the Nodei is the $HID_i^n$ calculated by $Un$, as shown in Equation (7), and for any $n' \neq n$, $HID_i^{n'} \neq HID_i^n$. In other words, at the initiate status of the protocol, only $Un$ knows the real identity $ID_i$ of Nodei, while $U_n$ only knows the pseudonym of Nodei, $HID_i^n$. Thus, for Nodei, the only way he can obtain $ID_i$ is to get $ID_i'\leftarrow HID_i^n$, making $HID_i = Hash[ID_i||Hash(D_n)]$. Due to the collision-resistance [41] nature of a cryptographic hash function, the probability that Nodei can find a $ID_i'$ which satisfies $HID_i = Hash[ID_i||Hash(D_n)] = Hash[ID_i'||Hash(D_n)]$ is negligible.

During the protocol execution, $ID_i$ does not participate in the agreement in any way and is not transmitted over the network, including encrypted-form or hashed-form, whereas $HID_i^n$ only participates in the identity declaration step, which is only used as a parameter to the one-way hash function in the computation of $SI_{i,n}^t$. Therefore, it is much more difficult for $\mathcal{A}$ or any others $Un$ to get $ID_i$ since the only $ID_i$-related information he could access during protocol execution is $SI_{i,n}^t$, which performs two independent hash operations using $HID_i^n$ as a parameter, and the result is XOR together as the argument of the function $f(\cdot)$. Thus, the probability that $\mathcal{A}$ and any others can get $ID_i$ by $SI_{i,n}^t$ is much smaller than the probability that $Un$ gets $ID_i$ form $HID_i$, which is negligible.

Consider the last case, where $Node_i$ or $\mathcal{A}$ try to guess each bit of $ID_i$, and get $ID_i'\leftarrow HID_i^n$. In this case, the probability $Pr[ID_i = ID_i']$ depends on the length of $ID_i$, that is, how many bits $ID_i$ has. For any single bit, the probability of guessing correctly is $1/2$, and for $n$ bits, $Pr[ID_i = ID_i'] = (1/2)^n$, which can be negligible if $ID_i$ has enough bits.

In conclusion, the probability that Nodei or $\mathcal{A}$ can obtain $ID_i$ is negligible, and thus the protocol satisfies the anonymity of Nodei.

An active attacker can steal $ID_i$ directly from Nodei or Un by executing Corrupt$(P)$ listing in the threat model defined in Section 5.1. When $P$ of Corrupt$(P)$ represent $Un$, $\mathcal{A}$ can only obtain the pseudonym $HID_i^n$, and $\mathcal{A}$ can obtain the real identity $ID_i$ of Nodei only if $P$ represent Nodei. As assumed in Section 5.1, we consider $Un$ to be secure, and thus $\mathcal{A}$ can only operate Corrupt$(P)$ to Nodei. Section 5.4.4 will further analyze the security of the scenario where Nodei was corrupted.

Now, consider the not-so-serious case where $\mathcal{A}$ does not use the Corrupt$(P)$ operation but, somehow, gets $ID_i$ leaked by $Node_i$ or $Un$. In this scenario, information obtained from the $ID_i$ is minimal for $\mathcal{A}$. $\mathcal{A}$ can get nothing about Nodei or other nodes through the $ID_i$, for which $ID_i$ is only a random value generated by $Un$ and not related to any information about the Nodei. $\mathcal{A}$ can neither get the Nodei’s location or runtime environment through $ID_i$, nor can he get the Nodei’s $ID_i$, through $ID_i$. $\mathcal{A}$ can use $ID_i$ to obtain $HID_i^n$ through Equation (7) since $D_n$ is public, and then he can calculate the $SI_{i,n}^t$ between Nodei and all subsequent actual controllers C through Equation (11). However, this will result in only a limited impact since C still cannot obtain the final $SI_{i,n}^t$, because the key information $q_i$ used for computing $SI_{i,n}^t$ is encrypted by $D_n$ and the attacker cannot decrypt it. More discussion about this serious can be found in Section 5.4.1. Also, $\mathcal{A}$ cannot insert a malicious node into the controlled group $G$ by fabricating an $ID_i'\leftarrow HID_i^n$, which is negligible. Similar to the anonymity of $Un$, the anonymity of $Un$ mainly refers to the security of $Un$’s identity. The identity of $Un$ is identified by the digital certificate $D_n$ issued by $Un$, which consists of two parts: the public key $D_n^{pub}$ and the private key $D_n^{priv}$. $D_n^{pub}$ is public and all nodes, including attackers, can easily obtain $D_n^{pub}$ from the $SI_{i,n}^t$ at the identity declaration step, while $D_n^{priv}$ is private and secured by $Un$. Due to the nature of asymmetric cryptography, it is feasible to calculate $D_n^{pub}$ from $D_n^{priv}$, $\mathcal{A}$ can guess $G_n^{priv}$ and make $G_n^{pub} = D_n^{pub}$. However, since private keys for asymmetric cryptography algorithms typically have a sufficient bit length, $Pr[G_n^{priv} = D_n^{pub}]$ is negligible. Similar to $ID_i$, $D_n$ only represents the control authority of $Un$ over the controlled group $G$ and has no connection with the real identity of $Un$, so $\mathcal{A}$ cannot obtain the real identity of $Un$ through the information in the protocol.

In addition, $D_n$ must contain a valid signature from $Un$ to be valid, so it is impossible that $\mathcal{A}$ can gain control of $Un$ by forging a $D_n$ without $CA_{priv}$. Each $D_n$ has a validity period, which means that in each identity declaration cycle, $Un$ needs to update $D_n$ and request a new signature from $Un$ for the updated $D_n$. In addition, $U_i$ has the authority to revoke any $Un$’s certificate by simply posting a certificate revocation request in $SI_{i,n}^t$.

5.4.4. Reverse Attack on Nodei. The different attack operations that an attacker $\mathcal{A}$ can perform are enumerated in the threat model in Section 5.1, in which ESReveal$(P)$, SKReveal$(P)$ and Corrupt$(P)$ imply that an attacker $\mathcal{A}$ can steal information from both parties involved in the protocol to varying degrees. Corrupt$(P)$ is the attack operation in which $\mathcal{A}$ can obtain the most information, and it is also the
The most serious attack faced by both participants involved in the protocol. Since we assume that $U_n$ and $U_r$ are secure in the attack model, the $ESReveal(P)$, $SKReveal(P)$, and $Corrupt(P)$ attacks target the controlled node $Node_i$, and $Corrupt(P)$ indicates that $\mathcal{A}$ can ultimately compromise $Node_i$ through reverse engineering or similar techniques and obtain all the sensitive data stored in $Node_i$ as well as the details of the protocol. In these attack scenarios, $\mathcal{A}$ may have access to $C_{ij}$, $ID_i$ of the compromised node $Node_i$, $SM_{ij}$, $K_{ij}$, as well as any information exchanged between $Node_i$ and $U_n$, $CA_{pub}$, and $D_{n}^{pub}$. However, $\mathcal{A}$ cannot get the ID $ID_i$ of the other node $Node_i'$ in $\mathbb{N}$ where $i' \neq i$, nor can it get any information related to the real identity of $U_n$, $U_r$, since $CA_{pub}$, $D_{n}^{pub}$ are all randomly generated. $\mathcal{A}$ is also unable to gain control of $G$ since $\mathcal{A}$ is unable to obtain or forge $CA_{priv}$ or $D_{n}^{priv}$, which are never transmitted over the network.

However, $\mathcal{A}$ does have methods to interfere with the execution of the protocol. $\mathcal{A}$ can interfere with the normal operation of $G$ by inserting a large amount of meaningless data into $SL_{i,j}'$ to interfere with the identity agreement process between $U_n$ and $\mathbb{N}$, since $SL_{i,j}'$ is wholly public and all nodes in $\mathbb{N}$ can send data to it. $\mathcal{A}$ can make $SL_{i,j}'$ full of garbage data so that $Node_i$ has to find the only valid data from a pile of garbage in $SL_{i,j}'$, which will reduce the possibility of a triumphant identity declaration and even threaten the security of $Node_i$, since $Node_i$ has to request $SL_{i,j}'$ numerous times in a short period, which may be a risk of deanonymization from the point of view of an anonymous communication system. A special warning mechanism can be used to handle the above situation. Specifically, $U_i$ can specify an alarm threshold $\xi$ for all $Node_i \in \mathbb{N}$, which represents the upper limit of the number of times $Node_i$ tries to negotiate identity using $SL_{i,j}'$. For any $C_j$, once $Node_i$ finds $\xi$ controller identity files $IF^c$ that do not contain legitimate digital signatures in $SL_{i,j}'$ cumulatively, $Node_i$ goes to the alert state, in which state nodes no longer request any data in $SL_{i,j}'$ but instead calculate a particular identity space $SL_{i,j}$, which is computed as follows:

$$SL_{i,j} = f(Hash(ID_i || K_j)).$$

From the computational procedure of $SL_{i,j}'$, it is clear that $SL_{i,j}$ is shared between $Node_i$ and $U_n$, and for $i \neq i'$, $SL_{i,j} \neq SL_{i',j}$. After calculating $SL_{i,j}'$, $Node_i$ first writes an alert message into $SL_{i,j}'$, then waits for $U_i$ to issue a special control instruction to $SL_{i,j}'$. Once getting control instructions, $Node_i$ performs a restore to normal state operation or an update operation according to $U_i$’s special control instruction. Meanwhile, when $U_i$ finds that some nodes are abnormally not negotiating their identity, or when $U_n$ also finds a large amount of false data in $SL_{i,j}'$, $U_n$ can request $U_i$ to check $SL_{i,j}'$. Once $U_i$ finds an early alert message sent by $Node_i$ in $SL_{i,j}'$, $U_n$ can infer that $G$ has anomalies. By the number of nodes sending alert messages, $U_n$ can evaluate the status of $G$ and indicate accordingly, e.g., let the node restore its normal state when $U_n$ considers this alert as a miscalculation or let the node perform an update operation on the acquisition of $K_i$ or the calculation of $SL_{i,j}$. The protocol using $SL_{i,j}'$ is too computationally expensive and inflexible, which requires the controller to compute an $SL_{i,j}'$ for each $Node_i$ and issue a copy of $IM_{i,j}'$ for every $SL_{i,j}'$. However, since $SL_{i,j}'$ of different nodes do not affect each other, it is appropriate to use $SL_{i,j}$ as an alternative when an attack against $SL_{i,j}$ is detected.

The above analysis shows that after $\mathcal{A}$ compromises a node, he can use the compromised node to interfere with the identity declaration phase, thus affecting the agreement protocol. However, $\mathcal{A}$’s influence on other nodes is limited, and according to the analysis in Section 5.4.1, $\mathcal{A}$ also cannot recover the expired session, so $\mathcal{A}$ cannot obtain the encrypted data that was previously transmitted.

5.4.5. Other Security Requirements. The security analysis in the previous section addresses the data steganography and identity anonymity of $U_n$, $U_r$, and $Node_i$, which the attacker tries to compromise in the threat model shown in Section 5.1. Other security requirements considered by the proposed protocol include the security of the control nodes, resilience to replay attacks, and the integrity of the transmitted data.

In the protocol, the identity of the controller $U_n$ is identified by the digital certificate $D_n$ issued for it by $U_r$, $Node_i$, only checks whether the identity file $IF^c_n$ issued by $U_n$ contains a legitimate $D_n$ and the digital signature $DS_n$, without caring which node $U_r$ uses to issue $IF^c_n$. In fact, in the Crowds system, $Node_i$ does not have access to information related to the reallocation of $U_n$ or the type of node used by $U_r$. In this way, the controller can use any node for control operations without being associated with a specific node; the only thing the controller needs is a legal $D_n$.

Replay attacks are meaningless for this protocol since all data generated within the protocol depends on parameters associated with $C_j$, such as $D_n$ or $K_j$. In addition, all data published to the Virtual-Space receives protection from cryptographic algorithms using digital signatures or asymmetric encryption, which means any tampering by an attacker will be immediately detected.

6. Implementation and Evaluation

Since there have been a large number of results on the comparison of Chebyshev polynomials with other cryptographic algorithms, such as [6, 13, 42], here we focus on evaluating the protocol designed in this paper at Freenet, which is named Hyphanet (https://www.hyphanet.org/index.html) now, on verifying the feasibility of the protocol. The prototype of the proposed protocol was written in C and Python, which uses OpenSSL (https://www.openssl.org) to implement cryptographic arithmetic operations, uses the algorithm of Zhang et al. [6] to compute Chebyshev polynomials, and uses PyFreenet3 (https://github.com/hyphanet/pyFreenet) library to communicate with Freenet. The prototype is running on a computer with an Intel(R) Core(TM) i5-12400 CPU and 32 GB of RAM, running Ubuntu 20.03, and the Freenet node is running on a cloud server powered by vultr (https://www.vultr.com) with 1 vCPU and 1,024 MB of RAM, running Ubuntu 22.04 LTS. The Critical Configuration of Freenet during the experiment is shown in Table 2.
The experiment runs two Freenet instances with the same configuration, keeping around 15 peers connected to each instance. The first instance is denoted as $I_1$, and its set of peers is denoted as $P_1$, and the second instance is denoted as $I_2$ with $P_2$ as its peers set. It is guaranteed that $P_1 \cap P_2 = \emptyset$. $I_1$ is labeled as a controller-operated node, and its position of Freenet is fixed to 0, while $I_2$ is labeled as a controlled node with a position from 0 to 0.5. Thus, the logical distance between $I_1$ and $I_2$ is 0 to 0.5, which covers the minimum logical distance 0 and maximum logical distance 0.5 possible in Freenet. Due to the extensive network fluctuations in Freenet, each set of experiments was conducted three times independently, and the results were averaged. The result of the experiments is shown in Figure 2.

The results show that the protocol elapsed time is minimally affected by the bit length of the Chebyshev polynomial when executing the protocol in practice Freenet since Freenet, as a medium-to-high latency anonymous file-sharing system, has file upload and download delays on the order of minutes, while the time consumption of the protocol on the order of milliseconds. Compared to the latency of network IO, the time consumed by the protocol can be negligible.

To realistically reflect the protocol computation time consuming, we remove all Freenet IO in the prototype at the end and convert the data upload/download operations to memory operations. The protocol runtime after removing Freenet IO operations is shown in Table 3. We ran the protocols 1,000 times for each length and got the average result. Since converting CPU clocks to milliseconds introduces additional error, we keep CPU clocks as units in our results.

### 7. Conclusion

In this work, we remedy the deficiencies of previous work by designing an identity-based Virtual-Space agreement method...
for a Virtual-Space-based remote control scheme based on Chebyshev polynomials. The protocol achieves two-way authentication between controlled and controlling nodes and Virtual-Space agreement for transmitting messages anonymously. The designed protocol supports independent Virtual-Space agreement of multiple controllers to multiple controlled nodes, and different nodes are free from each other. By conducting validation experiments on Freenet and performing a detailed security analysis, we demonstrate that the agreement method described in this paper can meet the security requirements of the Virtual-Space-based anonymous control scheme.

**Data Availability**

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

**Conflicts of Interest**

The authors have no conflicts of interest with regard to this work.

**Authors’ Contributions**

Kai Lin contributed in the writing and the methodology. Kaiyu Wang and Jin Shang contributed in the validation. Qindong Sun contributed in the paper framework and funding acquisition.

**Acknowledgments**

The research presented in this paper is supported in part by the National Natural Science Foundation (no.: 62272378), Shaanxi Province Key Research and Development Program (no.: 2022ZDLSF07-07), The Youth Innovation Team of Shaanxi Universities (no.: 2019-38), Natural Science Foundation of Sichuan Province (no.: 2023NSFSC0502), and Project of Xi’an Science and Technology Bureau (no. 22GXFW0079).

**References**


