Hindawi
Security and Communication Networks
Volume 2023, Article ID 5486566, 12 pages
https://doi.org/10.1155/2023/5486566

Research Article

Design and Implementation of a Lightweight Security-Enhanced Scheme for Modbus TCP Protocol

Zechao Liu,[1] **Tao Liang**,[1] **Wenshan Wang**,[1] **Ruochen Sun**,[1] **and Sizhao Li**[1,2]

[1]*Harbin Engineering University, Heilongjiang, Harbin 150001, China*
[2]*Modeling and Emulation in E-Government National Engineering Laboratory, Beijing 100037, China*

Correspondence should be addressed to Sizhao Li; sizhao.li@hrbeu.edu.cn

Received 27 October 2022; Revised 7 March 2023; Accepted 13 March 2023; Published 13 April 2023

Academic Editor: Irshad Azeem

The advent of Industry 4.0 has made people pay more and more attention to the security of the industrial control system. As a common and typical communication protocol, Modbus does not consider the problem of data security at the beginning of its design, which provides opportunities for criminals. In this paper, we design a security scheme to protect the traditional Modbus TCP protocol, by using domestic encryption algorithms. As a result, the proposed scheme is able to identity authentication, data encryption, data integrity check, and anti-replay attacks. Security analysis and experimental results show that our proposed scheme solves the security problem of Modbus TCP protocol with minimal overhead increase.

1. Introduction

With the continuous progress and innovation of technology, the Internet of Things (IoT) has been on the road of vigorous development. At the same time, more and more hackers exploit the vulnerabilities of the IoT to attack the network, resulting in the illegal leakage of a large amount of data and even paralyzing terminal [1]. This has brought serious losses and security risks to relevant enterprises and even countries. With the coming of Industry 4.0, using information technology to promote industrial change has been putting forward, and the Industrial Internet of Things (IIoT) has become an important branch of the IoT [2]. Industrial Control Systems (ICSs) play an important role in IIoT and are widely used in manufacturing, power generation, chemical manufacturing, sewage treatment, tobacco, and other industries [3]. An ICS is a group of industrial controllers and equipment to ensure the smooth functioning of an industrial facility, consisting of automated components that collect and monitor data in real time. The core components of ICS include fieldbus control system (FCS), Supervisory Control and Data Acquisition (SCADA) system, and distributed control system (DCS) [4, 5]. Since industrial control systems are carriers of data and instructions, they will face increasingly serious cyber threats, and the attacks of hackers on industrial control systems are endless. For example, in 2010, a hacker group used the Stuxnet virus to attack an Iranian nuclear enrichment plant, disabling more than 1,000 centrifuges [6]. In 2021, the municipal water treatment system in Oldsmar, Florida, was hacked in an attempt to raise the concentration of sodium hydroxide to dangerous levels. Clearly, ICS security is closely related to national security and people's quality of life.

In order to deal with different application scenarios in the industrial control system, many communication protocols are designed for ICS, such as Modbus, Distributed Network Protocol 3 (DNP3), and Controller Area Network Bus (CANBUS). Among them, as a serial communication protocol, Modbus has become one of the most popular communication protocols due to its simple and easy-to-understand messages format and open standard [7]. Modbus TCP protocol is a communication mode of Modbus protocol, which is an Ethernet application-layer communication protocol based on TCP/IP. It is widely used in industrial control equipment produced by Siemens, Schneider, Mitsubishi, and other industrial control manufacturers, as well as intelligent manufacturing, energy, power, and other industries. However, the traditional Modbus TCP protocol has

great security risks. Attackers can not only intercept the plaintext data during the communication between Modbus TCP client and Modbus TCP server but also maliciously tamper with these data [8]. In addition, attackers can connect directly to the Modbus TCP server, send illegal commands, or get data from the Modbus TCP server.

The State Cryptography Administration of China has published some domestic cryptographic algorithm standards and their application specifications to improve the country's autonomous controllability in information security. The algorithm involves a variety of encryption types, including public key encryption algorithms SM2 and SM9; symmetric encryption algorithms SM1, SM4, and SM7; and hash algorithm SM3. The popularization of the national cryptographic algorithms is of great significance to improve China's network information security and the autonomous control level [9]. In this paper, we propose a security enhancement scheme of Modbus TCP mainly through the domestic encryption algorithms SM3 and SM4, which can improve the security of Modbus TCP.

In this work, we propose a security-enhanced scheme for Modbus TCP protocol. The main contributions in this paper are summarized as follows:

(1) We propose a security scheme of Modbus TCP protocol based on domestic cryptographic algorithms. Our scheme includes authentication, key negotiation, and secure data transmission. First, in the authentication and key negotiation phase, we complete the mutual authentication and key negotiation between the client and server by preshared key and the SM4 algorithm. Second, in the secure data transmission phase, we add the timestamp and messages authentication code fields to the traditional Modbus TCP messages. The timestamp field is used to prevent replay attacks, and the messages authentication code field is used to detect the integrity and legitimacy of Modbus TCP messages. Finally, we encrypt key data through the SM4 algorithm to ensure the data security.

(2) We analyse the security of our scheme, and the analysis shows that the protocol security problem solved by the scheme proposed in this paper is more comprehensive compared with other security schemes.

(3) We perform an experimental evaluation of the Modbus TCP security scheme proposed in this paper, and the results show that the proposed scheme has advantages in both computation overhead and storage overhead compared with other security schemes.

2. Related Work

In recent years, there are some related works on the security of Modbus TCP. Shahzad et al. [10] added extra security fields to standard Modbus packets and used RSA and AES algorithms to generate keys and encrypt data, respectively. Shang et al. [11] proposed an industrial firewall design based on Modbus TCP protocol, which combined "whitelist" with firewall technology to effectively intercept illegal data streams and ensure the normal operation of industrial control systems. Pricop et al. [12] proposed authentication through the hash algorithm and put the authenticated data in the option field of TCP header. Alves et al. [13] designed the intermediary agent OpenPLC. It encrypts the data transmitted by ICS protocol through the AES-256 algorithm to ensure the confidentiality of data. However, this scheme does not provide authentication. Xuan and Yongzhong [14] proposed the Modbus-S scheme to ensure the credibility of the identity of the communication parties and the confidentiality of the data. Jingran et al. [15] applied TLS technology to Modbus TCP protocol. However, the digital certificates involved in TLS increase the computation overhead and administrative overhead significantly. Yi et al. [16] proposed the Modbus TCP security scheme to achieve data encryption and integrity verification and prevent replay attacks by timestamp and random numbers. Nevertheless, since the information is transmitted in plaintext, its anti-replay ability is weak. Lin et al. [17] proposed a Modbus security scheme based on the AES algorithm and SHA-256 hash algorithm, but this scheme cannot provide authentication. Chochtoula et al. [18] proposed a Modbus security scheme based on elliptic curve cryptography, providing methods for key generation, key negotiation, and data encryption and decryption.

3. Domestic Cryptographic Algorithms

3.1. SM3 Hash Algorithm. SM3 is a hash algorithm independently developed in the core field of cipher in China. For a message m with L ($L < 2^{64}$) bits, the SM3 hash algorithm generates a hash value with 256 bits after messages filling, iterative compression.

(1) Fixed parameters

IV = 7380166f 4914b2b9 172442d7 da8a0600 a96f30bc 163138aa e38dee4d b0fb0e4e

$$
\begin{aligned}
T_j &= \begin{cases} 79cc4519, 0 \le j \le 15, \\ 7a879d8a, 16 \le j \le 63, \end{cases} \\
FF_j(X,Y,Z) &= \begin{cases} X \oplus Y \oplus Z, 0 \le j \le 15, \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), 16 \le j \le 63, \end{cases} \\
GG_j(X,Y,Z) &= \begin{cases} X \oplus Y \oplus Z, 0 \le j \le 15, \\ (X \wedge Y) \vee (X \wedge Z), 16 \le j \le 63, \end{cases} \\
P_0(X) &= X \oplus (X \ll 9) \oplus (X \ll 17), \\
P_1(X) &= X \oplus (X \ll 15) \oplus (X \ll 23).
\end{aligned} \tag{1}
$$

(2) Messages filling

Assuming the message m is L bits long, first add the bit "1" to the end of the messages and then add k zeros, k being the smallest non-negative integer satisfying $L + 1 + k \equiv 448 \pmod{512}$. It then adds a 64-bit bit string, which is the binary representation

of length L. The length of the message is the multiple of 512 bits.

(3) Iterative compression

Iterative process

Group the filled messages m' by 512 bits, $m' = B^{(0)}B^{(1)} \dots B^{(n-1)}$, where $n = (L + k + 65)/512$. Iterate over m' as follows:

FOR $i = 0$ TO $n - 1$,

$$V^{(i+1)} = \mathrm{CF}\left(V^{(i)}, B^{(i)}\right). \tag{2}$$

ENDFOR

$V^{(0)}$ is the initial value IV of 256 bits, and CF is the compression function.

Messages extension

The messages group $B^{(i)}$ is extended to generate 132 messages words $W_0, W_1 \dots\dots W_{67}, W_0' \dots \dots W_{63}'$ and is used to compress the function CF.

(a) The messages group $B^{(i)}$ is divided into 16 words $W_0 \dots\dots W_{15}$

(b) FOR $j = 16$ TO 67

$$W_j \longleftarrow P_1\left(W_{j-16} \oplus W_{j-9} \oplus \left(W_{j-3} \lll 15\right)\right) \oplus \left(W_{j-13} \lll 7\right) \oplus W_{j-6}. \tag{3}$$

ENDFOR

(c) FOR $j = 0$ TO 63

$$W_j' = W_j \oplus W_{j+4}. \tag{4}$$

ENDFOR

Compression function

A, B, C, D, E, F, G, and H are word registers

FOR $j = 0$ TO 63

$$\begin{aligned}
SS1 &\longleftarrow \left((A \lll 12) + E + (T_i \lll (j \bmod 32))\right) \lll 7\\
SS2 &\longleftarrow SS1 \oplus (A \lll 12)\\
TT1 &\longleftarrow FF_j(A, B, C) + D + SS2 + W_j'\\
TT2 &\longleftarrow GG_j(E, F, G) + H + SS1 + W_j\\
D &\longleftarrow C\\
C &\longleftarrow B \lll 9\\
B &\longleftarrow A\\
A &\longleftarrow TT1\\
H &\longleftarrow G\\
G &\longleftarrow F \lll 19\\
F &\longleftarrow E\\
E &\longleftarrow P_0(TT2).
\end{aligned} \tag{5}$$

ENDFOR

$$V^{(i+1)} \longleftarrow ABCDEFGH \oplus V^{(i)}. \tag{6}$$

(4) Hash value

$$ABCDEFGH \longleftarrow V^{(n)}. \tag{7}$$

The final output is the 256 bit hash value $y = ABCDEFGH$.

3.2. SM4 Symmetric Encryption Algorithm. SM4 is a symmetric encryption algorithm independently developed in China. The SM4 algorithm is a block encryption algorithm with the key length and block length of 128 bit. It includes 32 iteration operations and one reverse order transformation R.

Suppose that the plaintext input is $(X_0, X_1, X_2, X_3) \in (Z_2^{32})^4$, the ciphertext output is $(Y_0, Y_1, Y_2, Y_3) \in (Z_2^{32})^4$, and the round key is $rk_i \in Z_2^{32}$, $i = 0, 1, 2, \dots, 31$, the round function: $F(X_0, X_1, X_2, X_3, rk) = X_0 \oplus T(X_1 \oplus X_2 \oplus X_3 \oplus rk)$, and T is the synthetic permutation function.

The operation process of the SM4 encryption algorithm is as follows:

(1) 32 iterations

$$X_i + 4 = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i), i = 0, 1, 2 \dots\dots 31. \tag{8}$$

(2) Reverse order transformation

$$\begin{aligned}
R(X_{32}, X_{33}, X_{34}, X_{35}) &= (X_{35}, X_{34}, X_{33}, X_{32}),\\
(Y_0, Y_1, Y_2, Y_3) &= R(X_{32}, X_{33}, X_{34}, X_{35}).
\end{aligned} \tag{9}$$

The SM4 decryption algorithm has the same structure as the encryption algorithm, but the difference is the sequence in which the round key is used. When decrypting, the round key sequence is $(rk_{31}, rk_{30}, \dots\dots, rk_0)$.

4. Modbus TCP Protocol

4.1. Modbus TCP/IP Network Topology. Modbus is a communication protocol based on master-slave architecture. It has three communication protocols: Modbus RTU protocol, Modbus ASCII protocol, and Modbus TCP protocol. Among them, Modbus RTU protocol and Modbus ASCII protocol are based on serial links to communication. Modbus TCP protocol is based on Ethernet to communication.

As a branch of Modbus protocol, Modbus TCP protocol consists of two entities: Modbus TCP client and Modbus TCP server. The client corresponds to Modbus master, and the server corresponds to Modbus slave. Modbus TCP/IP network topology is shown in Figure 1. In the network, only the client sends request to the server, and then, the server can respond to the client. Otherwise, the server cannot take the initiative to send messages to the client. The Modbus TCP/IP network not only allows clients and servers to connect

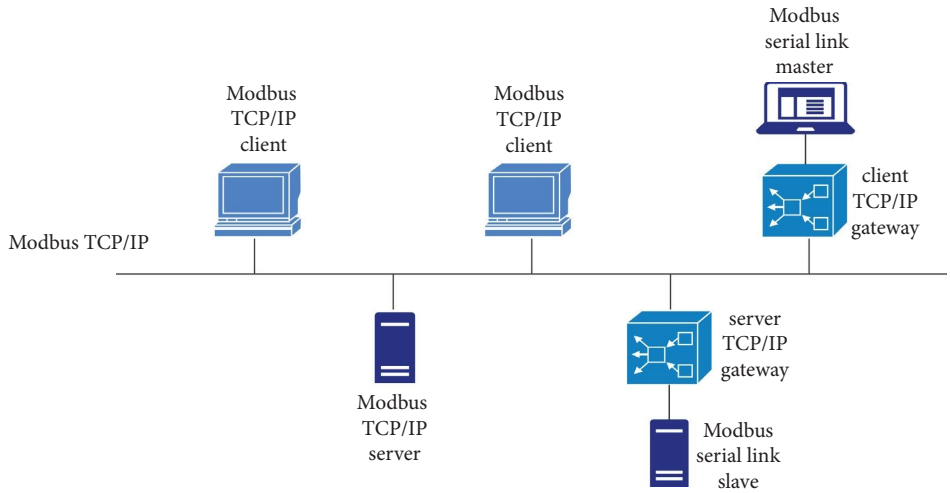


Figure 1: Modbus TCP protocol network topology.

directly but also allows Modbus masters and slaves in serial links to connect through routers, gateways, and other devices.

4.2. Modbus TCP Messages Structure. The messages structure of Modbus TCP protocol is shown in Figure 2 [19].

In Modbus TCP protocol, the function code field and the data field are collectively called protocol data unit (PDU). Modbus application protocol (MBAP) messages headers and PDU are collectively called application data unit (ADU).

The MBAP messages header field is seven bytes, including transaction identifier, protocol identifier, length, and unit identifier. Specifically, transaction identifiers take two bytes, which are used to mark different communication messages, ensure message accuracy, and prevent message confusion across the network. The protocol identifier contains two bytes, namely, 00 00, which indicates Modbus TCP protocol. The length field is two bytes, which stores the length of unit identifier, function code, and data. The unit identifier field takes up one byte and stores the address of Modbus TCP server or the serial link device connected to the Modbus TCP/IP network.

The PDU is a field for storing key data, consisting of the function code and data. The function code occupies one byte and represents the operation the Modbus TCP client wants to perform on the Modbus TCP server. Common function codes are shown in Table 1.

5. Modbus TCP Security-Enhanced Scheme

The symbol definitions used in this section are shown in Table 2.

5.1. Authentication and the Key Negotiation Phase. In this phase, the Modbus TCP server can identify the legitimacy of the connected Modbus TCP client, and the Modbus TCP client can identify whether it is connected to the target server or not. In the authentication phase, the preshared key PSK agreed between the client and server is used. If the authentication succeeds, the session key negotiation is based on the parameters passed during the authentication. The specific steps of authentication and key negotiation are described as follows.

(1) The Modbus TCP client generates a string of 128 bits random number R_m and calculates H_{m1} and C_m. Then, the client sends C_m to the Modbus TCP server.

$$\begin{cases} H_{m1} = H(\mathrm{PSK}R_m), \\ C_m = E_{\mathrm{PSK}}(\mathrm{PSK}R_m H_{m1}). \end{cases} \tag{10}$$

(2) The Modbus TCP server receives the message and decrypts C_m by the preshared key PSK. Then, it gets the random number R_m by XOR operation between the local *PSK* and the first half of the decrypted message and finally computes H_{s1}. If H_{s1} is equal to H_{m1}, it proves the legitimate identity of Modbus TCP client. Otherwise, the message is discarded.

$$\begin{aligned} \mathrm{PSK}R_m H_{m1} &= D_{\mathrm{PSK}}(C_m), \\ H_{s1} &= H(\mathrm{PSK}R_m). \end{aligned} \tag{11}$$

After the Modbus TCP client passes the authentication, the Modbus TCP server generates a random number R_s with 128 bits and then calculates H_{s2} and C_s. Next, the server sends C_s to the client.

$$\begin{cases} H_{s2} = H(R_m R_s), \\ C_s = E_{\mathrm{PSK}}(R_m R_s H_{s2}). \end{cases} \tag{12}$$

(3) The Modbus TCP client receives the message and decrypts C_s by the preshared key PSK.

$$R_m R_s H_{s2} = D_{\mathrm{PSK}}(C_s). \tag{13}$$

It can obtain the random number R_s by performing XOR operation between local random number R_m and the first half of the plaintext. Then, the Modbus TCP client computes H_{m2}.

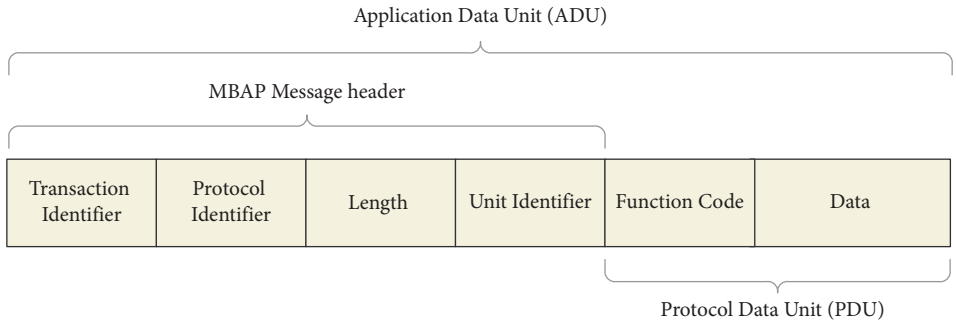


Figure 2: Modbus TCP application data unit structure.

Table 1: Various Modbus TCP function codes and their actions on data.

Function codes	Action to be performed
01	Reading coils
02	Read discrete input
03	Read holding registers
04	Read input registers
05	Write single coils
06	Write single register
0F	Write multiple coils
10	Write multiple registers

Table 2: Symbol used in the proposed scheme.

Symbols	Description
PSK	Preshared key between the client and the server
R_m	Random number generated by the client
R_s	Random number generated by the server
Key	Session key negotiated between the client and the server
$\|\|$	Connect function
$E()$	SM4 encryption
$D()$	SM4 decryption
$H()$	SM3 hash algorithm
$H()_{16}$	The first 16 bytes of the SM3 hash value
C_m	Ciphertext encrypted by the client using preshared key
C_s	Ciphertext encrypted by the server using preshared key
H_{m1}, H_{m2}, H_{m3}	Hash value generated by the client
H_{s1}, H_{s2}, H_{s3}	Hash value generated by the server
C_{PT}	Ciphertext encrypted by the negotiated session key

$$H_{m2} = H(R_m R_s). \tag{14}$$

The legitimate identity of the Modbus TCP server can be proved, and the integrity of random number R_s is verified if H_{m2} is equal to H_{s2}. Otherwise, the message are discarded. After the Modbus TCP server passes the authentication, the Modbus TCP client generates the session key for the data transfer phase. And then, the client sends "Verify_successful" to the server to indicate that he has been authenticated.

$$\mathrm{Key} = H\left(R_m \| R_s \| \mathrm{PSK}\right)_{16}. \tag{15}$$

(4) After receiving the message "Verify_successful," the Modbus TCP server uses the same algorithm to generate the session key "Key."

5.2. Secure Data Transmission Phase. In the phase of secure data transmission, the SM4 algorithm is used to encrypt important data into ciphertext with the session key generated in the previous phase. Transmitting ciphertext prevents the attacker from obtaining the plaintext of important data. However, if the attacker knows the function of each message, it can replay the message even if it does not know the plaintext. Therefore, the timestamp field is added in the traditional Modbus TCP to prevent the replay attack. The SM3 hash algorithm is used to ensure the integrity of data during transmission and prevent data from being tampered illegally. The Modbus TCP protocol data format of security-enhanced is shown in Figure 3.

The specific steps of the secure data transmission phase are described as follows. The operation flowchart of the sender (Modbus TCP client) during the secure data transmission phase is shown in Figure 4.

(1) The Modbus TCP client obtains the current system time and adds the timestamp into the timestamp field of the secure Modbus TCP messages.

(2) The Modbus TCP client uses the SM4 algorithm to encrypt the function code, data, and timestamp fields by session key "Key" and puts the ciphertext into the corresponding location of the security messages.

$$C_{PT} = E_{\mathrm{Key}}(\mathrm{Function\,Code} \| \mathrm{Data} \| \mathrm{Timestamp}). \tag{16}$$

(3) The Modbus TCP client takes the C_{PT} and the session key "Key" as the input of the SM3 hash algorithm, generates the hash value H_{m3}, and puts it to the messages authentication code field.

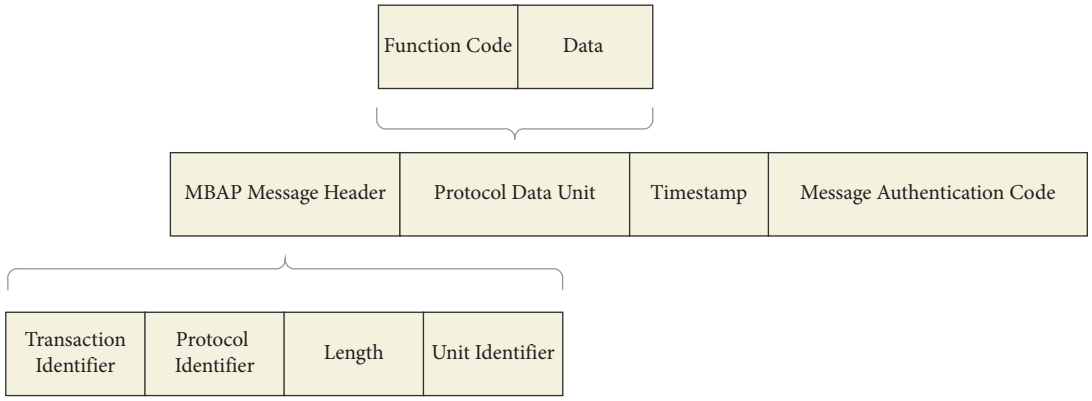


Figure 3: Modbus TCP protocol data format of security-enhanced.

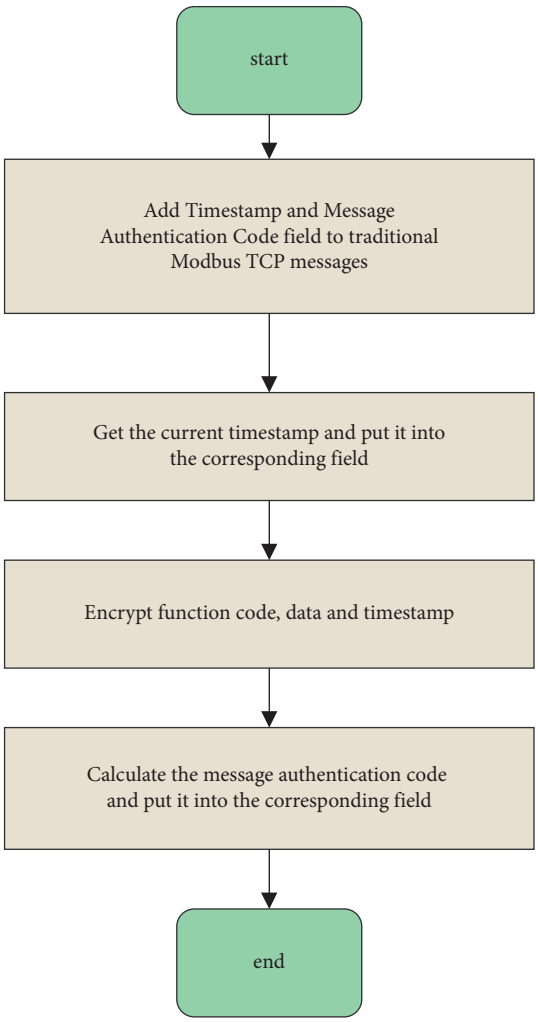


Figure 4: The operation flow of the sender.

$$H_{m3} = H(C_{PT}\text{Key}). \tag{17}$$

Figure 5 shows the operation flowchart of the receiver (Modbus TCP server) in the secure data transmission phase.

(1) The Modbus TCP server takes the ciphertext C_{PT} and session key "Key" as the input of the SM3 hash algorithm to generate the hash value H_{s3} and compares it with the content in the received messages authentication code field. If they are the same, it can prove that the source of the messages is credible and the messages cannot be tampered maliciously. Otherwise, the messages will be discarded.

$$H_{s3} = H(C_{PT}\text{Key}). \tag{18}$$

(2) The Modbus TCP server decrypts C_{PT} through the session key "Key" to get timestamp. Then, it generates the timestamp by the system time and compares it with the timestamp received. If the timestamp is within the valid range, the Modbus TCP server performs operations based on function code and data. Otherwise, the messages will be discarded.

$$\text{Function Code}\|\text{Data}\|\text{Timestamp} = D_{\text{Key}}(C_{PT}). \tag{19}$$

6. Security Analysis

6.1. Authentication and the Key Negotiation Phase. In this phase, the Modbus client and the Modbus server use preshared keys and the SM4 algorithm for mutual authentication and key negotiation. The connection will be disconnected if the authentication fails. Since the transmitted message is encrypted and hidden through XOR operation and the SM4 algorithm, even if the attacker intercepts the message, they cannot get the information they want.

To improve the security of the session key, the scheme generates random numbers every time the Modbus TCP client connects to the Modbus TCP server and then negotiates the final session key. The session key negotiated in this phase is used for encryption and decryption the critical data in the next phase. This ensures that the session keys are different each time the Modbus TCP client connects to the Modbus TCP server.

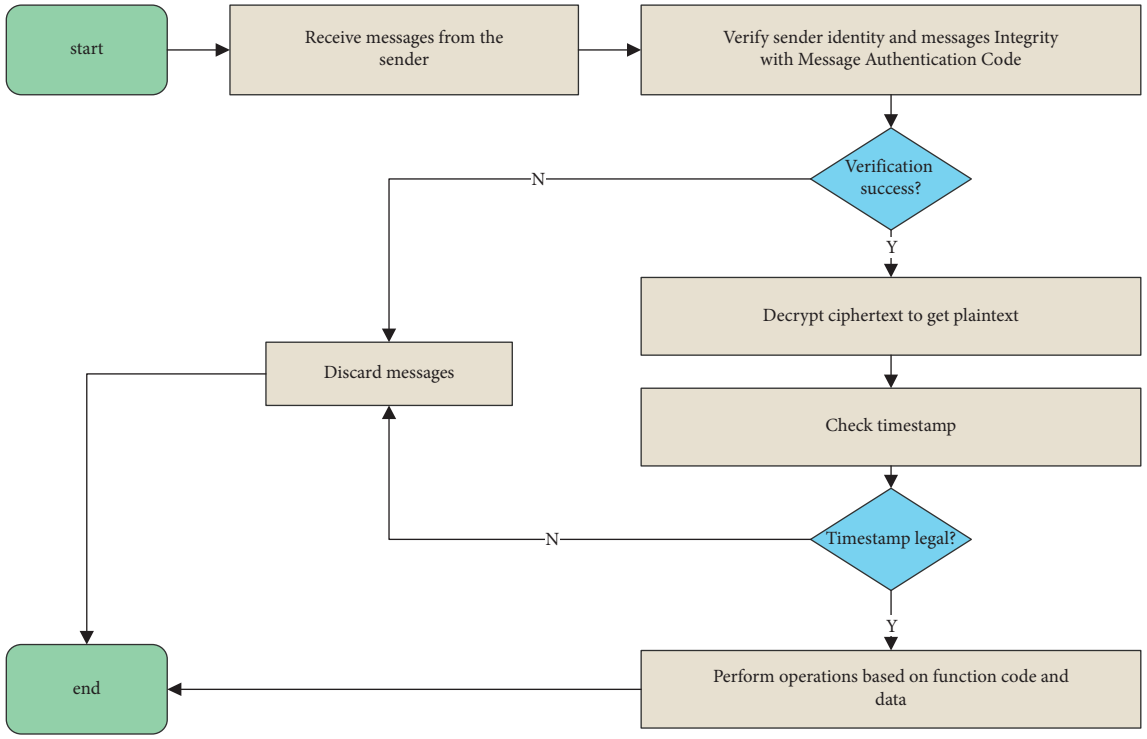


Figure 5: The operation flow of the receiver.

6.2. Secure Data Transmission Phase. In this scheme, two fields of timestamp and messages authentication code are added to the traditional Modbus TCP messages, making them be the Modbus TCP secure messages.

First, make sure the identity of the sender is legitimate and the message is not maliciously tampered with. On the one hand, the session key "Key" is one of the inputs of the message authentication code, and only the terminal that passes the authentication and key negotiation phase can obtain the key. Therefore, the message authentication codes calculated by the two parties are different if the identity of the sender is illegal. On the other hand, C_{pt} is another input to the message authentication code, where C_{pt} is the ciphertext of the function code, data, and timestamp. Since the attackers cannot know the session key, if they maliciously tamper with C_{pt}, they cannot calculate the same message authentication code. This will result in the received message authentication code different from the calculated message authentication code for the receiver. Therefore, as long as the message authentication codes are the same, the sender's legitimacy and the integrity of the message can be proved.

Second, the current timestamp is stored in the timestamp field of the Modbus TCP security messages to prevent replay attacks. The receiver compares it to its own system timestamp each time it receives the messages. If the timestamp is within the safe range, it will perform operations based on the function code and data. Because the timestamp is encrypted, the attackers cannot modify the timestamp, even if the messages are intercepted. If the attackers send the same messages more than once, the timestamp will be out of the safe range.

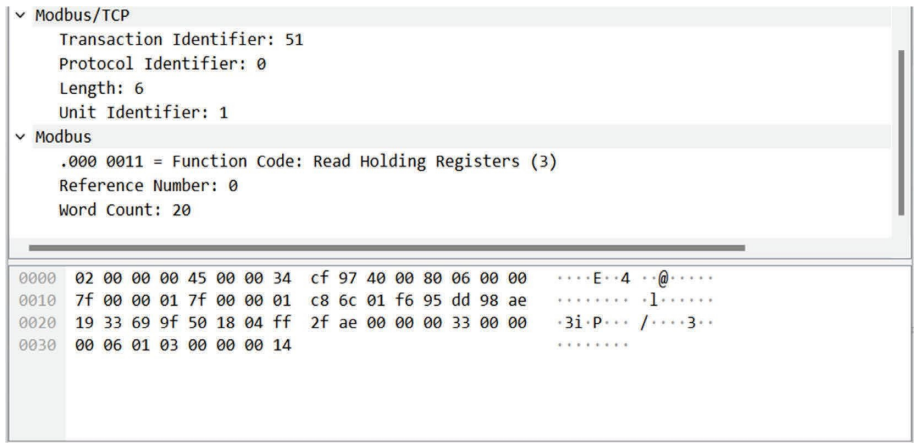
Third, the proposed scheme uses the SM4 symmetric encryption algorithm to encrypt the function code, data, and timestamp at the same time, which ensures that the transmitted key data are not accessed illegally by attackers.

6.3. Comparison of Modbus TCP Security. To highlight the security of this scheme, this section compares it with other Modbus TCP security enhancement schemes, as shown in Table 3.

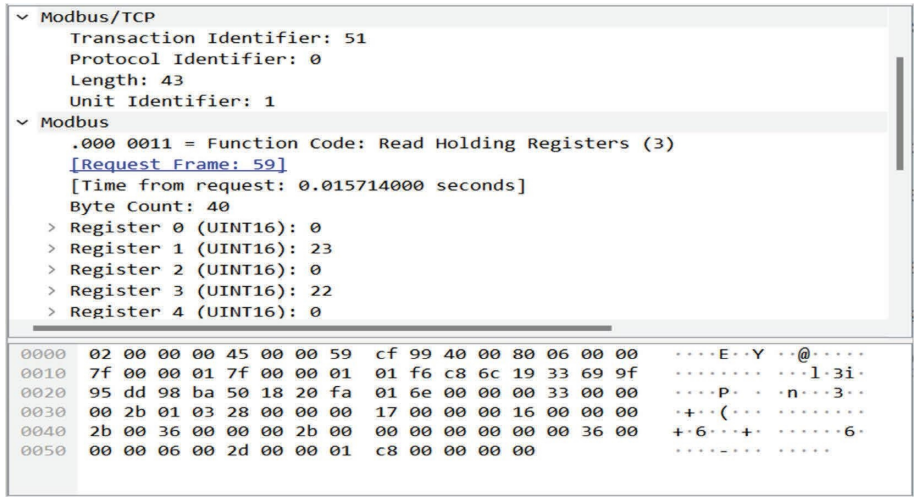
The schemes [10, 14] use the AES symmetric encryption algorithm to encrypt the data in Modbus TCP protocol and use the RSA algorithm to sign the data to achieve identity authentication. Scheme [10] uses the SHA-2 algorithm to calculate the hash value, while scheme [14] uses the MD5 algorithm to calculate the hash value to verify the integrity of the messages. The scheme [14] prevents replay attacks by computing synchronization identifiers, but scheme [10] does not propose a mechanism to prevent replay attacks. Scheme [15] applies TLS to Modbus TCP protocol to realize data encryption, authentication, data integrity check, anti-replay attack, and key update mechanism. Scheme [17] uses the AES symmetric encryption algorithm to encrypt the data in Modbus TCP protocol, uses the SHA-256 algorithm to calculate the hash value, and verifies messages integrity and prevents replay attacks through timestamps. Scheme [16] uses the SM4 symmetric encryption algorithm to encrypt the data in Modbus TCP protocol, calculates the hash value through the SM3 algorithm, verifies the integrity of the messages, and realizes identity authentication. It protects against replay attacks by random numbers and timestamp.

Table 3: Comparison of the security of the proposed scheme with related schemes.

Schemes	Identity authentication	Data encryption	Messages integrity check	Anti-replay	Key update
[10]	√	√	√	×	×
[14]	√	√	√	√	×
[15]	√	√	√	√	√
[16]	√	√	√	√	×
[17]	×	√	√	√	×
Our scheme	√	√	√	√	√



(a)



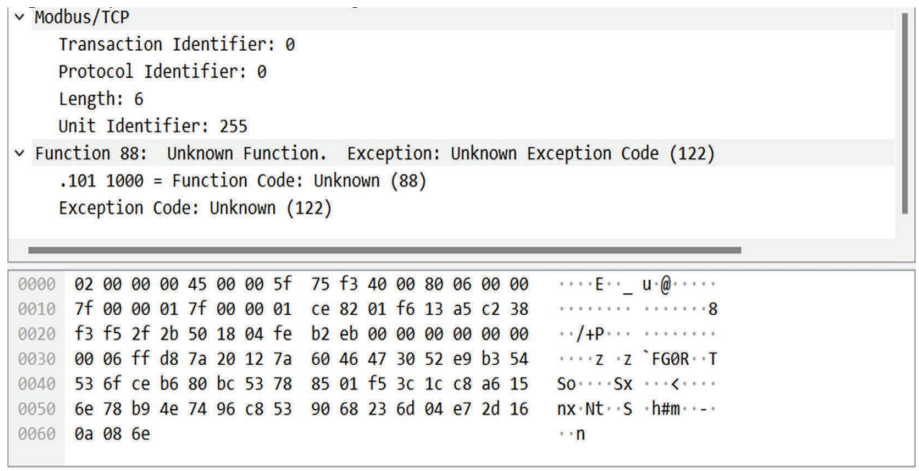
(b)

Figure 6: Message content for traditional Modbus TCP. (a) Sent by the Modbus TCP client. (b) Sent by the Modbus TCP server.

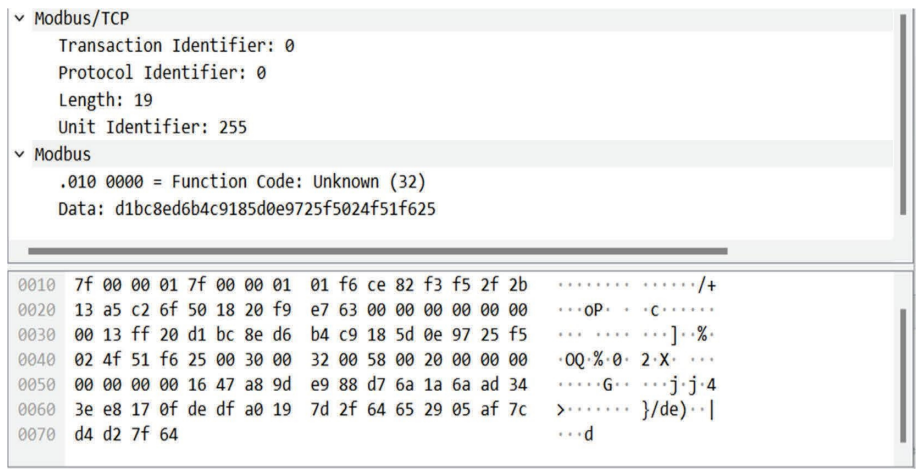
However, the random numbers and timestamps in this scheme are transmitted in plaintext, and the attacker can easily modify them and send them again after interception. Therefore, the scheme is less effective in preventing replay attacks. In contrast, our proposed scheme uses timestamp to prevent replay attacks, which not only does not need to open new memory space to store random numbers but also encrypts the timestamps for transmission. The attacker cannot modify timestamp data, which can prevent replay attacks effectively.

7. Experimental Results and Analysis

In this section, experiments are performed on the proposed scheme in terms of both computation overhead and storage overhead. To ensure the accuracy of the data, we did ten experiments to test the computation overhead. The scheme is divided into the authentication and key negotiation phase and the secure data transmission phase. Therefore, the computation overhead of these two stages is calculated separately in the experiment. All experiments are running on



(a)



(b)

Figure 7: Message content for security Modbus TCP. (a) Sent by the Modbus TCP client. (b) Sent by the Modbus TCP server.

Intel(R) Core (TM) i5-12500H@3.1 GHz CPU, 16 GB RAM, Windows 11 64 bit OS.

7.1. Experimental Results. We use Wireshark to intercept traditional Modbus TCP messages and encrypted Modbus TCP messages, respectively, to clearly see the effect of the proposed scheme during the secure data transmission phase. Function code 03 is used as an example here. Figure 6(a) shows the traditional Modbus TCP messages sent by the client, and Figure 6(b) shows the traditional Modbus TCP messages sent by the server. It can be seen from Figure 6 that data are transmitted in plaintext during traditional Modbus TCP communication, which means that attackers can easily obtain key information such as the function code and data and tamper with it maliciously. Hence, the protocol is insecure at this time.

Figure 7(a) shows the security messages sent by the Modbus TCP client of the proposed scheme, where the function code and data have been encrypted. The last 32 bytes of the data captured by the Wireshark is the messages authentication code.

The Modbus TCP server receives the messages and returns the corresponding messages after authentication, integrity check, and timestamp check. Figure 7(b) shows the data returned by the Modbus TCP server. In this figure, the critical data have been encrypted. Wireshark only intercepts some nonsensical message sequences. This means that attackers cannot obtain the plaintext of critical data and tamper with it maliciously.

7.2. Computation Overhead. To ensure the accuracy of the experimental data, we conducted ten experiments for each phase separately and then calculated the average value. Numbers in Figures 8 and 9 represent the number of experiments.

In the authentication and key negotiation phase, Figure 8(a) shows the computation overhead of the Modbus TCP client, while Figure 8(b) shows the computation overhead of the Modbus TCP server. Experimental results show that the average computation overhead of Modbus TCP client is 11.71 μs. The average computation overhead of the Modbus TCP server is 9.76 μs.

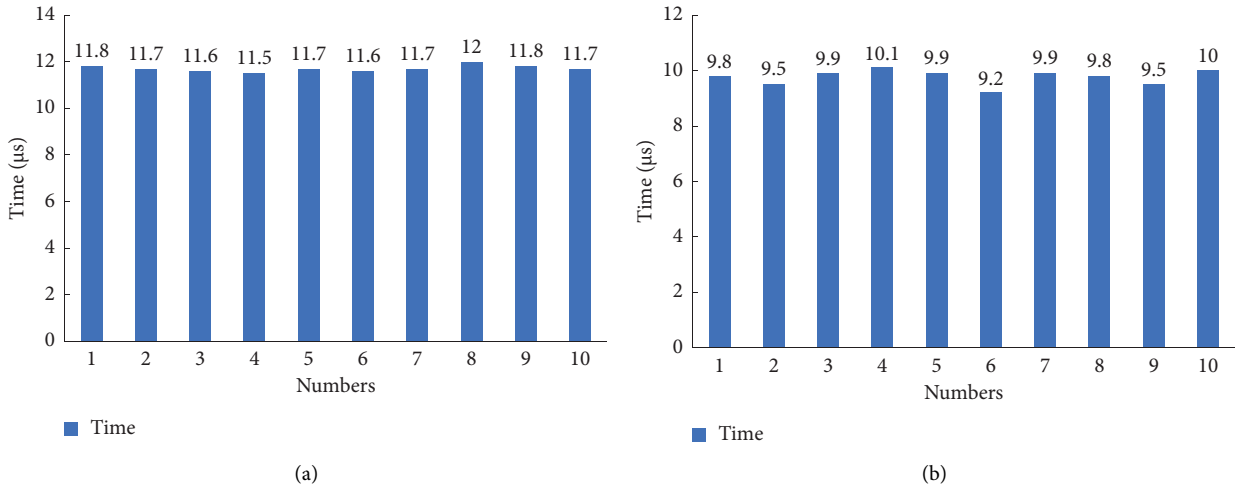


Figure 8: Computation overhead in authentication and key negotiation phase. (a) Modbus TCP client. (b) Modbus TCP server.

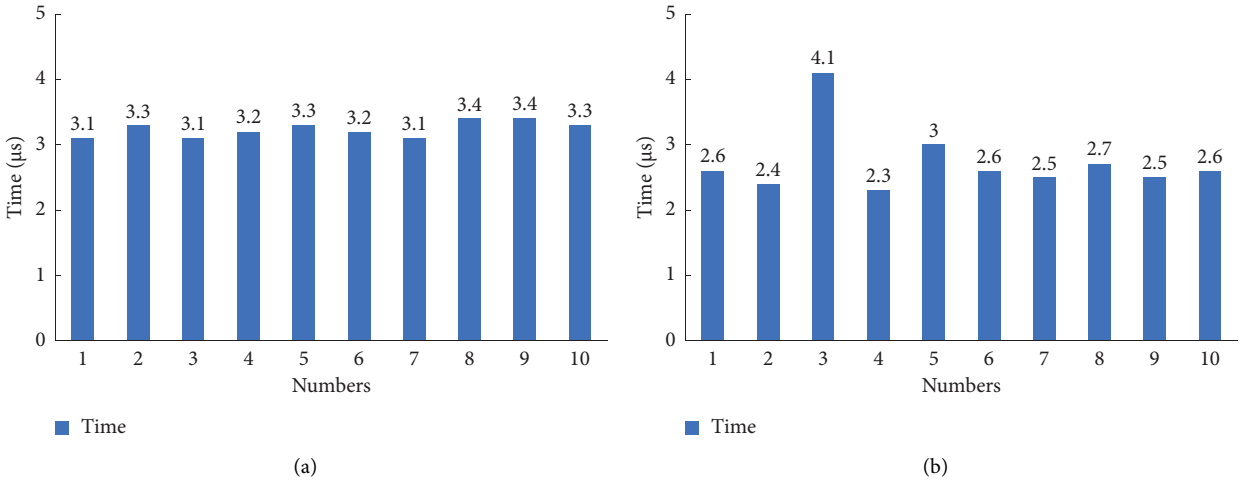


Figure 9: Computation overhead in the secure data transmission phase. (a) Modbus TCP client. (b) Modbus TCP server.

In the secure data transmission phase, the timestamp and messages authentication code are calculated and the key data of Modbus TCP messages are encrypted or decrypted.

Figure 9(a) shows the computation overhead of the Modbus TCP client, while Figure 9(b) shows the computation overhead of the Modbus TCP server. Experimental results show that the average computation overhead of the Modbus TCP client is 3.24 μs and the average computation overhead of the Modbus TCP server is 2.73 μs in the secure data transmission phase.

In Table 4, we show the performance comparison between our scheme and other related works. Some symbols are defined as follows. H denotes a hash operation, AES_E denotes one AES encryption operation, AES_D denotes one AES decryption operation, $SM4_E$ denotes one SM4 encryption operation, $SM4_D$ denotes one SM4 decryption operation, RSA_S denotes one RSA signature operation, RSA_V denotes one RSA signature verification operation, and TLS denotes a complete TLS protocol process.

The public key encryption algorithm is an encryption algorithm which uses complex mathematical problems as security guarantee. The encryption and decryption operations and signature verification operations contain many mathematical operations. Therefore, its computation efficiency is much lower than that of the symmetric encryption algorithm and the hash algorithm. The RSA algorithm used in schemes [10, 14] is a public key encryption algorithm that uses large integer factorization mathematical problems as security guarantees. The overhead of RSA is much higher than that of AES, SM4, and hash algorithms. Scheme [15] applies TLS protocol to Modbus TCP protocol to effectively ensure security. But the TLS protocol involves authentication of digital certificates and some RSA operations, which causes the computation overhead to increase exponentially. Schemes [16, 17] have slightly lower computation overhead. The computation overhead of schemes [16, 17] is slightly better than that of the scheme in this paper. However, security analysis shows that scheme [16] does not realize key

Table 4: Comparison of computation overhead with related schemes.

Schemes	Computation overhead (sender)	Computation overhead (receiver)
[10]	$H + AES_E + RSA_S$	$H + AES_D + RSA_V$
[14]	$2H + AES_E + RSA_S$	$2H + AES_D + RSA_V$
[15]	TLS	TLS
[16]	$2H + SM4_E$	$2H + SM4_D$
[17]	$H + AES_E$	$H + AES_D$
Our scheme	$4H + 2SM4_E + SM4_D$	$4H + SM4_E + 2SM4_D$

Table 5: Comparison of storage overhead with related schemes.

Schemes	Storage overhead
[10]	3072 bits + AESkeys_size
[14]	3072 bits + AES_size + seed_size + WL_size
[15]	2certificate_size + 2304 bits
[16]	128 bits + RL_size
[17]	256 bits
Our scheme	256 bits

update and has anti-replay ability is weak. Scheme [17] lacks the authentication mechanism. To sum up, the proposed scheme is worthy of higher security with less performance loss.

7.3. Storage Overhead. We compare the storage overhead with other Modbus TCP security enhancement schemes, and the comparison results are shown in Table 5. Note that "∗_size" represents the storage size of "∗." Schemes [10, 14] adopt the RSA algorithm for signature verification and the AES algorithm for encryption and decryption. Therefore, both parties need to store their RSA public and private keys and AES symmetric keys. Scheme [14] also needs to store a seed and a whitelist WL. The TLS protocol of scheme [15] involves the verification of digital certificates, so it needs to store its own digital certificate and own RSA (2048) private key and the other party's digital certificate. The storage size of each digital certificate is about 1 KB~3 KB. In addition, the encryption algorithm used in scheme [15] is ChaCha20-poly1305. Therefore, the key of the algorithm also needs to be stored. Scheme [16] adopts the SM4 algorithm for encryption and decryption, and both communication parties need to store the SM4 symmetric key. Scheme [16] also needs to store a table RL for storing random numbers, in order to achieve the purpose of anti-replay attack. Scheme [17] has the same storage overhead with our proposed schemes. But the security of our proposed scheme is better than scheme [17].

8. Conclusion

In this paper, we design a Modbus TCP security mechanism based on domestic cryptography algorithms. First, we complete mutual authentication and session key negotiation through the preshared key, SM4 encryption algorithm, XOR operation, and so on. Then, we design Modbus TCP security message which adds anti-replay mechanism through the timestamp field and calculate the message authentication code through the SM3 algorithm to verify the integrity and legitimacy of the message. Finally, we encrypt key data such as the function code, data, and timestamp through the SM4 algorithm. The security analysis and experimental results show that the proposed scheme can protect Modbus TCP autonomously and controllably with smaller computation overhead and storage overhead.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the National Key R & D Program of China (no. 2021YFB3101602) and Basic Research Program (no. JCKY2020604C011).

References

[1] S. Bhatt and P. R. Ragiri, "Security trends in Internet of Things: a survey," *SN Applied Sciences*, vol. 3, no. 1, pp. 1–14, 2021.

[2] Y. Li, Q. Cheng, and W. Shi, "Security analysis of a lightweight identity-based two-party Authenticated key agreement protocol for IIoT environments," *Security and Communication Networks*, vol. 2021, Article ID 5573886, 6 pages, 2021.

[3] Y. Jiang, L. Wang, and X. Zhang, "Tobacco system industrial control system security," in *Proceedings of the IEEE 4th International Conference on Computer and Communication Systems*, pp. 693–696, Singapore, February 2019.

[4] Z. Song, A. Skuric, and K. Ji, "A recursive watermark method for hard real-time industrial control system cyber-resilience enhancement," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 1030–1043, 2020.

[5] C. Zhou, X. Li, S. Yang, and Y. C. Tian, "Risk-based scheduling of security tasks in industrial control systems with consideration of safety," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3112–3123, 2020.

[6] A. Nourian and S. Madnick, "A systems theoretic approach to the security threats in cyber physical systems applied to stuxnet," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 2–13, 2018.

[7] V. G. Găitan and I. Zagan, "Experimental implementation and performance evaluation of an iot access gateway for the modbus extension," *Sensors*, vol. 21, no. 1, p. 246, 2021.

[8] S. Yan-feng, W. Shao-jie, and Q. Yu, "Summary of security technology and application in industrial control system," *Computer Science*, vol. 45, no. 4, pp. 25–33, 2018.

[9] L. Zhang and Y. Ge, "Identity authentication based on domestic commercial cryptography with blockchain in the heterogeneous alliance network," in *Proceedings of the IEEE International Conference on Consumer Electronics and Computer Engineering*, pp. 191–195, IEEE, Guangzhou, China, January 2021.

[10] A. Shahzad, M. Lee, Y. K. Lee et al., "Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information," *Symmetry*, vol. 7, no. 3, pp. 1176–1210, 2015.

[11] W. Shang, Q. Qiao, M. Wan, and P. Zeng, "Design and implementation of industrial firewall for Modbus/TCP," *Journal of Computers*, vol. 11, no. 5, pp. 432–438, 2016.

[12] E. Pricop, J. Fattahi, and N. Parashiv, "Method for authentication of sensors connected on modbus tcp," in *Proceedings of the The 4th International Conference on Control, Decision and Information Technologies*, pp. 0679–0683, IEEE, Barcelona, Spain, April 2017.

[13] T. Alves, T. Morris, and S. M. Yoo, "Securing scada applications using openplc with end-to-end encryption," in *Proceedings of the 3rd Annual Industrial Control System Security Workshop*, pp. 1–6, Orlando, Fl, USA, December 2017.

[14] L. Xuan and L. Yongzhong, "Research and implementation of Modbus TCP security enhancement protocol," *Journal of Physics: Conference Series*, vol. 1213, no. 5, Article ID 052058, 2019.

[15] W. Jingran, L. Mingzhe, and X. Aidong, "Research and implementation of secure industrial communication protocols," in *Proceedings of the IEEE International Conference on Artificial Intelligence and Information Systems*, pp. 314–317, IEEE, Kota Kinabalu, Malaysia, September 2020.

[16] F. Yi, L. Zhang, and S. Yang, "A security-enhanced Modbus TCP protocol and authorized access mechanism," in *Proceedings of the IEEE 6th international conference on data science in cyberspace*, pp. 61–67, IEEE, Shenzhen, China, October 2021.

[17] Y. C. Lin, C. F. Lin, and K. H. Chen, "Security enhancement of industrial Modbus mess-age transmission with proxy approach," in *Proceedings of the IEEE 3rd eurasia conference on IOT, communication and engineering*, pp. 90–95, IEEE, Yunlin, Taiwan, October 2021.

[18] D. Chochtoula, A. Ilias, Y. C. Stamatiou, and C. Makris, "Integrating elliptic curve cryptography with the Modbus TCP SCADA communication protocol," *Future Internet*, vol. 14, no. 8, p. 232, 2022.

[19] F. R. Ametov, E. A. Bekirov, and M. M. Asanov, "Organizing the information security in Modbus TCP interfaces for use in the energy complex," *IOP Conference Series: Materials Science and Engineering*, vol. 1089, no. 1, Article ID 012007, 2021.