Diffie-hellman Protocol on Raspberry pi

Yuwen Wang and Gabriela Mogos*
School of Advanced Technology, Xi’an-Jiaotong Liverpool University, Suzhou, China

*Corresponding author email: gabriela.mogos@xjtlu.edu.cn

Abstract. Cryptography is a technique that applies the number theory to maintain information security. Securing data often means encrypting it. Each of the known cryptosystems has advantages and disadvantages. The most popular key exchange protocol is the Diffie-Hellman protocol. The paper presents the obtained results of the Diffie-Hellman protocol’s implementation on portable devices, namely on raspberry boards.

1. Introduction
The Internet is an extremely open environment, anyone can access or transmit information at any time and in any place, which makes the information security a hot spot of today’s society.
The research of information security includes cryptography theory and technology, security protocol and technology, security architecture theory [1], information confrontation theory and many other fields.
The development and implementation of cryptographic algorithms are the foundation of information security research. Also, the reliability and efficiency of data encryption plays a vital role in the application of algorithm theory to various security products.
Cryptography offers security solutions for all types of information: in use, in transit and in store.
There can be no cryptography without a cryptographic key. Cryptosystems are mainly divided into three: symmetric key encryption, asymmetric key encryption, and mixed key encryption. The symmetric cryptosystem uses the same key for encryption and decryption. The distribution of the cryptographic key is one of the major problems of this cryptosystem. The security of obtaining the cryptographic key is one of the major problems of the symmetric cryptosystem. For this reason, a high-size cryptographic key is used, which causes data processing to be very slow.
The asymmetric key encryption solves the major problem of symmetrical algorithms. Two keys will be used: a public key and a private key for encryption and decryption. Because the public key and the private key are not the same, it will be difficult for an attacker to decrypt an encrypted message. Even if the size of the cryptographic key is reduced the cryptosystem is robust. Finally, for hybrid encryption, symmetric key encryption and asymmetric key encryption are used together to ensure the security and transmission efficiency.
Most of the current key agreement protocols are based on the Diffie-Hellman key transmission protocol. The paper presents the obtained results by the implementation of the Diffie-Hellman protocol on raspberry systems. Given the limitations of raspberry systems, the project has tracked the maximum size of the key that can be obtained on such devices. No cyber-attacks were simulated.

2. Implementation

2.1. Diffie-Hellman Protocol
The Diffie-Hellman key distribution protocol is the first key exchange negotiation scheme proposed by
Whitefield Diffie and Martin Hellman in 1976 [2]. The Diffie-Hellman key transmission protocol algorithm is an asymmetric key agreement. Its goal is to allow two users to exchange keys securely, even they use a communication channel under cyber-attacks. Let consider A and B, the parties involved in DH protocol. They will generate pairs of keys (public key - private key) that will then be used for encryption and decryption. It should be noted that the algorithm itself is not used for encryption and decryption, but only for the establishment of keys and exchange of keys by the two parties on an insecure channel. It is to be known that key generation does not require the parties to exchange information in advance, as they are vulnerable to a man-in-the-middle attack.

2.1.1. Protocol validity. The effectiveness of Diffie-Hellman Algorithm depends on the difficulty of calculating discrete logarithms. An attacker can get information about p, g, YA, YB, but for KA or Kg’s calculation needs XA or XB. Taking the calculation of XA as an example, the attacker needs to obtain XA according to the conditions YA = gXA mod p. In fact, when p is a large prime number, it is quite difficult. The difficulty of the discrete logarithm problem solves the safety and efficiency of the DH algorithm.

2.1.2. Protocol security. The protocol security is ensured by the size of the cryptographic key. Due to the discrete logarithm problem, if the number of prime numbers p is infinite, it is difficult to calculate XA, XB from known conditions. The primitive root g does not need to be very large. Generally, the primitive root with the lowest prime number is selected, usually select 2 or 5. The vulnerability of protocol to man-in-the-middle attacks is known.

2.2. Raspberry pi
In this project, two raspberry pi 4b were selected for system design. Raspberry pi 4b is the latest product now. Compared to the previous version, the raspberry pi 4b has a more powerful processor, faster read and writes speed, and greatly improves its performance. The CPU of the raspberry pi 4b is a quad-core Cortex-A72 with a maximum frequency of 1.5GHz and a GPU frequency of 500MHz. The RAM of raspberry pi 4b is 2 GB. In addition, the network transmission rate has also been greatly improved, using a proprietary RGMII link to achieve a transmission speed close to the Gigabit network.

![Figure 1. Raspberry pi 4b.](image)

The data transmission and processing in the two systems can be divided into three ways. The first is to store the data in the database, and continuously perform operations such as storage calculation, query update, and deletion. The system continuously accesses the database for real-time data update. But the database needs to ensure the independence of the data, the data and the program are independent of each other, and no more redundant data should appear. The second is to connect the two devices through the line, so that the two devices can access each other to exchange data.
The third is through TCP/IP [5] and other protocols, through the connection of IP addresses and ports for data transmission on the Internet. After the connection is established on the two devices, data transmission is performed, and the interface is closed after the transmission is completed.
In our project, the client IP address is 192.168.1.41; the server IP address is 192.168.1.40, and the port used in the communication is 9004.

![Figure 2. Server connection.](image)

![Figure 3. Client connection.](image)

### 2.3. Client-Server Application
One raspberry pi serves as the server and the other serves as the client. If the user wants to negotiate with the server using the Diffie-Hellman algorithm to obtain a public key, they need to interact and calculate through the client.
The specific steps are as follows:

**Step 1**: Open the server to prepare for a TCP connection and wait for the client to respond.  
**Step 2**: The client connects with the server and waits for the user to enter the number $n$ of prime $p$. The client then generates an $n$-digit prime $p$ and calculates the primitive root $g$ of the prime $p$ to generate the client's private random number $X_A$ is the customer's private key and calculates the public key $Y_A = g^{X_A} \mod p$. The client transmits the prime number $p$, the primitive root $g$ [3] of the prime number $p$, and the public key $Y_A$ to the server.  
**Step 3**: After receiving the data, the server generates the private random number $X_B$ of the server, which is the private key of the server. The server calculates the public key $Y_B = g^{X_B} \mod p$ and shared key $K_B = (Y_A)^{X_B} \mod p$ and sends the public key $Y_B$ and shared key $K_B$ to the client.  
**Step 4**: The client receives the data, calculates the client's shared key $K_A = (Y_B)^{X_A} \mod p$, and judges whether $K_A$ is equal to $K_B$. If it is equal, the calculation and transmission are successful this time. All the obtained data and the running time of this transmission are displayed on the client interface.  
**Step 5**: After the client has repeatedly requested to exchange keys, it can choose to close the program and disconnect from the server.  
At this point, the client and the client have completed the protocol exchange and obtained the required shared key.  
Diffie-Hellman key agreement algorithm is implemented in python.
3. Results
There were conducted four tests to reduce errors due to randomness, and the number and running time of each test were recorded. When the number of digits is 20, the program has been running for 15 minutes and the result has not been calculated, so the recording is stopped.

Table 1. Test results.

| Digit | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     | 10    |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| test1 | 0.4600| 0.4908| 0.6560| 0.4995| 0.8221| 0.6580| 0.5058| 1.3456| 0.6936| 1.8654|
| time(s)| 0.3353| 0.5594| 0.4981| 0.7537| 0.4710| 0.5010| 0.4304| 0.5821| 0.7125| 0.4574|
| test3 | 0.4566| 0.4600| 0.6447| 0.6178| 0.4347| 0.6753| 0.6688| 0.5992| 0.5258| 0.9070|
| time(s)| 0.4594| 0.3350| 0.6015| 0.5990| 0.5455| 0.5582| 0.5864| 0.4815| 0.6018| 0.5823|

Table 2. Test results.

| Digit | 11    | 12    | 13    | 14    | 15    | 16    | 17    | 18    | 19    |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| test1 | 0.5769| 1.7501| 1.1457| 2.7566| 2.2538| 7.2816| 42.0232| 40.4642| 472.9716|
| time(s)| 0.6881| 1.0856| 2.4165| 2.3881| 2.8615| 16.4920| 25.1701| 123.0103| 515.2743|
| test3 | 0.7405| 0.5859| 1.0339| 3.4211| 1.6990| 16.1961| 30.3622| 313.9441|       |
| time(s)| 0.7147| 1.0218| 0.9033| 1.8693| 2.4884| 11.8684| 31.9533| 201.4588| 333.0374|
The data is made with images to visually observe the relationship between time and digits, and the results of the four experiments are expressed in four colours. In the chart, the horizontal axis represents digits, and the vertical axis represents elapsed time. The chart is shown below:

![Graphic representation of data.](image)

**Figure 5.** Graphic representation of data.

According to the chart, the running time in the early stage was steadily maintained at 0.2s-3s, and the time required after 15 digits began to increase, and each additional 1 bit increased the amount of time increased significantly. After 16 digits, the running time increases exponentially with the increase of digits.

4. Conclusions

The project uses two raspberry pi as users who need to perform the Diffie-Hellman key agreement. Communication was based on the TCP/IP socket protocol and were considered a safe communication channel and there is no man-in-the-middle attack. To generate the large prime numbers the Miller-Rabin algorithm was used primitive root of prime numbers, the Euler's theorem was used. The application counts the number of digits of the key and the running time. The results show that the system can withstand up to 19 digits keys, but the running time increases exponentially after 15 digits keys.

References

[1] Payne R, Brown P, Ormston C, & Garrett A 2005 *Information security*, US.
[2] Diffie W, Hellman M 1999 *Diffie-Hellman Key Agreement Method*.
[3] Surhone L M, Timpledon M T & Marseken S F 2010 *Primitive root modulo n*.
[4] Corrigan-Gibbs H and Kogan D 2018 *The Discrete-Logarithm Problem with Preprocessing*.
[5] David A, Kassimis C and Stevens J W 2018 *Protocol selection for transmission control protocol/internet protocol (TCP/IP)*.