

# A SECURE COMMUNICATION PROTOCOL TO SUPPORT CLIENT AND SERVER COMPUTING

**Liangjun You**

Information Systems and Operations Management Department  
University of Texas at Arlington  
Arlington, TX 76019  
(817)272-3562  
[lyou@uta.edu](mailto:lyou@uta.edu)

**Dennis C. Guster**

Department of Business Computer Information Systems  
St. Cloud State University  
St. Cloud, MN 56301  
(320)229-5756  
[dcguster@stcloudstate.edu](mailto:dcguster@stcloudstate.edu)

## ABSTRACT

Currently, fast and inexpensive network communication and computational capacities provide humans with tremendous convenience and affordability in communicating and managing information. However, any information transmitted across the internet is in danger of being compromised. This paper proposes a prototype Secure Communication Protocol to Support Client/Server Computing. By adopting symmetric and asymmetric encryption techniques the client and the server can authenticate each other, the password is well protected from being stolen by the hackers, and a secure communication channel is set up to conduct the communication.

**Keywords:** secure protocol, computer networks, communication, client and server

## INTRODUCTION

As social creatures, people are born communication prone. And it is the communication process that makes a society operate. However people are also privacy sensitive. Although they are curious about business of other people, they tend to be adamant about safeguarding their own privacy. As stated “the urge to discover secrets is deeply ingrained in human nature (Chadwick, 1990).”

Currently, fast and inexpensive network communication and computational capacities provide humans with tremendous convenience and affordability in communicating and managing information. The number of networked computers is still increasing. And the IP address resources provided by IPV4, which are around  $2^{32}$ , are about to be depleted. However, while enjoying the great advantages of networking, people also realize that their privacy and security is endangered. Hacking, cracking and problems in network security draw more and more of people’s attention and have become a great concern to many people. From the frequency of

cracking, it is clear that any information transmission across the internet is in danger of being compromised.

In the development of network applications the most commonly used model is the client/server (or distributed system) model. This model is so widely used that more and more applications need access to a secure channel to conduct safe client/server communication. Many of the currently used applications and protocols are vulnerable to the hackers and interceptors on the internet. For instance, in UNIX systems services like telnet, ftp and rlogin, email and http are especially vulnerable. The ftp services hosted by Windows system are not safe either. They offer no encryption so the communication that takes place while a user logs in by providing the user I.D. and password is easily intercepted by a would-be hacker or opponent on the internet and the legitimate users' resources are in danger.

This paper suggests a Secure Communication Prototype Protocol to Support Client/Server Computing and produces the diagram to implementation it accordingly. This protocol insures mutual authentication and sets up a secure communication channel between client and server. Its logic can be widely used in a variety of client/server models serving various network applications.

This protocol will attempt to correct the drawbacks of the following scenarios:

1. In a client and server environment, when a client logs in onto the server, usually the client provides his User I.D. and Password, and then this chunk of information is passed via the media to the server in the clear. Under these circumstances, opponents can intercept the User I.D. and Password easily and then use this information to access the legitimate user's resources on the server. Further more, if the client and server cannot authenticate each other then the client may be tricked in logging onto a counterfeit server.
2. In some cases when an asymmetric encryption and decryption algorithm is adopted, the passing of the public key is in the clear, which still leaves some clues for potential opponents. The protocol this paper suggested cannot overcome such scenarios as the crackers break into systems without using user I.D. or password, etc. If you lose the password, it is just like you lose the key to your door, you are on your own. To operate the keyboard in a dark and electrical screened cubical can improve the overall computer security. But all these are out of the scope of this paper. This protocol can only prevent hackers or crackers from stealing password from the network. Hackers with unlimited access can still break in using brute force, so access control is imperative as far as password management is concerned.

## **BACKGROUND INFORMATION**

“ON A DAY nearly 4,000 years ago, in a town called Menet Khufu bordering the thin ribbon of the Nile, a master scribe sketched out the hieroglyphs that told the story of his lord's life—and in so doing he opened the recorded history of cryptology. His was not a system of secret writing, as the modern world knows it; he used no fully developed code of hieroglyphic symbol substitutions. His inscription carved about 1900 B.C into the living rock in the main chamber of the tomb of the nobleman Khnumhotip II, merely uses some unusual hieroglyphic symbols here and there in place of the more ordinary one. The intention was not to make it hard to read the text. It was to impart a dignity and authority to it, perhaps in the same way that a government proclamation will spell out “In the year of Our Lord One thousand eight hundred and sixty three”

instead of just writing “1863.” The anonymous scribe may also have been demonstrating his knowledge for posterity. Thus the inscription was not secret writing, but it incorporated one of the essential elements of cryptography: a deliberate transformation of the writing. It is the oldest text known to do so.” (Kahn, 1996)

Examples indicate that there were proto cryptographic systems created by the ancient Assyrians, Babylonians, Egyptians and Hebrews. In its first 3000 years, cryptology developed unsteadily. But there were still sparks of development a little now and a little then. One example was the Caesar cipher, which was improved and used by the Roman ruler Julius Caesar (100 B.C.–44 B.C.). It adopted a shift of three positions so that the plaintext A would be encrypted as D. For example,

Plaintext: ~~WHICH KEY IS ABOVE~~ UHH

There are only 25 possible keys for Caesar’s cipher.

Another cipher method adopts the transposition technique. In this technique, a paper stripe with the width of that of a character is used, the stripe is then wrapped around a thin cylinder as show in Figure 1. Then the message is written on the stripe longitudinally along the cylinder.

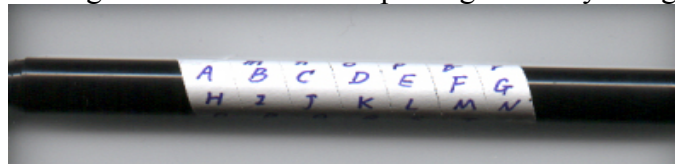


Figure 1: An Example of a Transposition Encryption Technique

The key for this cipher is the diameter of the cylinder. Wrapping the paper stripe around a cylinder of the same diameter can decrypt the cipher. Figure 2 shows the cipher message and the encryption-decryption cylinder.



Figure 2: The Transposition Encryption Result

These ciphers had already made use of the one of or both of the two general principles of cryptology: substitution, in which each element in the plaintext is replaced by other element according to specified rules, and transposition, in which the elements in the plaintext is rearranged according to specified rules. These techniques are classified as the classical encryption techniques. Obviously these primitive techniques use a small key size and they are easy to break.

Since the Western Renaissance, the development of cryptography becomes much faster than ever. More and more cryptographic techniques appeared and the ciphers became more complex. Such as cipher disk, shown in Figure 3, invented by the Father of Western Cryptology, Leon Battista Alberti (A.D.1404–A.D.1472) involved polyalphabetic substitution. This technique can hinder the analysis of frequencies used by cryptanalysis.

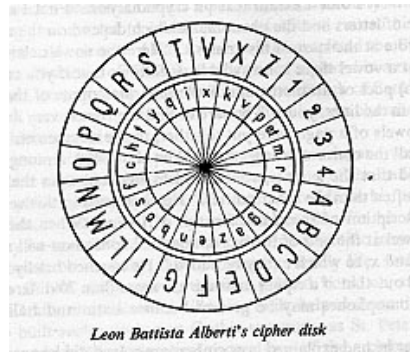


Figure 3 Cipher Disk (Kahn, 1996)

In the 1970s, Data Encryption Standard (DES), on which the most widely use encryption scheme based on, came into being. It uses bit-wise substitution and transposition, thus encrypting 64-bit blocks by using a 56-bit key. It adopts three Permutation Tables, one Permutation Function, eight Substitution-Boxes, and 16 rounds of permutations with different sub-keys derived from the original key. In this way, there are many more elements besides just the key and the plaintext involved in this encryption and decryption process. It takes 1142 years for a one-encryption/ $\mu$ s machine to break DES with brute force attack. In 1997, it took 96 days for personal computers distributed over the Internet, which at the end grew up to more than 70,000 systems, to find the DES key for the challenge cipher issued by RSA Laboratories. Not long ago, the DES Challenge III was solved by the Electronic Frontier Foundation's "Deep Crack" in a combined effort with distributed system with 100,000 PCs in less than 23 hours. However, it may only take only 10.01 hours for a  $10^6$ -encryptions/ $\mu$ s machine to break DES. The DES key size has to be increased periodically so as to make it stronger. The DES decryption is the reverse process of the encryption using the same key. Because the same key is used in the encryption and decryption process, DES is one of the techniques classified as symmetric cryptography. There are other symmetric such as International Data Encryption, RC5, etc. DES and the like are classified as modern techniques of conventional encryption.

The appearance of the public-key cryptography greatly enriches the cryptography family. This cryptology, classified as asymmetric, uses two different keys, one for encryption and the other for decryption. It does not use substitution or transposition any longer. The Rivest-Shamir-Adleman (RSA) scheme (Rivest, et al, 1978) is the most widely used, general-purpose approach to public-key encryption. RSA applies an expression with exponentials of:

$$M^{ed} \bmod n = M \bmod n, \quad (1)$$

where  $M$  is the plaintext block, and  $n$  is an integer known by the encrypting side and decrypting side, and  $e$  and  $d$  are the integer keys for encryption and decryption respectively. When encryption is conducted, the equation of

$$C = M^e \bmod n \quad (2)$$

is used. When decryption is conducted, the equation of

$$M = C^d \bmod n \quad (3)$$

is used.

It is infeasible to determine  $d$  given  $e$  and  $n$  and also it is relatively easy to compute  $M^e \bmod n$  and  $C^d \bmod n$  (Stallings, 2006). When the repeated squaring method is used to compute  $M^e \bmod n$  and  $C^d \bmod n$  and  $a$ ,  $b$ , and  $n$  are  $\beta$ -bit numbers, the total number of arithmetic operation is  $O(\beta)$  and the total number of bit operation is  $O(\beta^3)$  (Cormen, 1997).

Public-key cryptography is widely used in secret key distribution for symmetric cryptography and authentication. There are other public-key cryptographic algorithms, such as Diffie-Hellman algorithm, Elliptic Curve algorithm, etc. Generally speaking compared with symmetric cryptography, public-key cryptography has bigger time complexity.

File Transfer Protocol (FTP) is one of the most commonly used client/server protocols to transfer files. The FTP involves two connections between the client and the server as shown in Figure 4. One is the control connection, which is used to transmit the control commands. The other is the data connection, which is for the transmission of files. After the control connection is established, the USER NAME, PASSWORD, and ACCOUNT are sent to the server side via the control connection respectively. If the server decides that the user is a legitimate one, the data connection is established and files can be transmitted between the client and the server. But all the information transmitted by either connection is vulnerable since nothing it is encrypted.

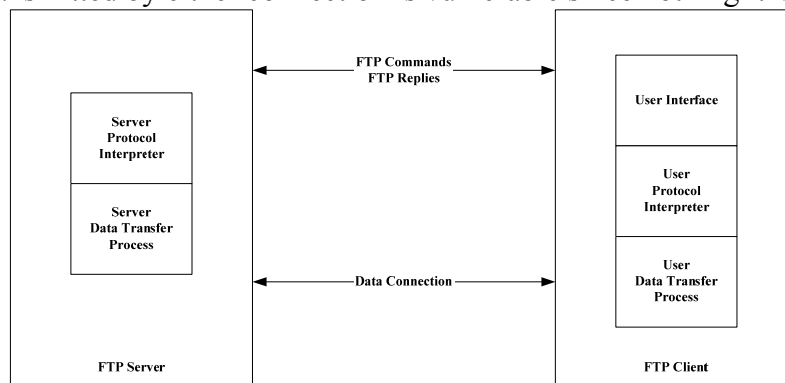


Figure 4: Connections for FTP

Based on the encryption and decryption algorithms, diverse secure services and protocols appeared. Merkle (Merkle, 1979) suggest a simple scheme to distribute secret key as depicted in Figure 5. When Alice wants to communicate with Bob in a secure way, first Alice generates a public/private key pair  $\{K_{Ua}, K_{Aa}\}$  and then sends the  $K_{Ua}$  and her I.D. to Bob. Once Bob receives the message, he generates a secret key and then sends the encrypted secret key by Alice's  $K_{Ua}$  back to Alice. Since only Alice knows the  $K_{Ua}$ , she can recover the secret key by decrypting the incoming message with the  $K_{Ua}$ . After that they all know the secret key and then they can use this secret key to set up a secure communication channel.

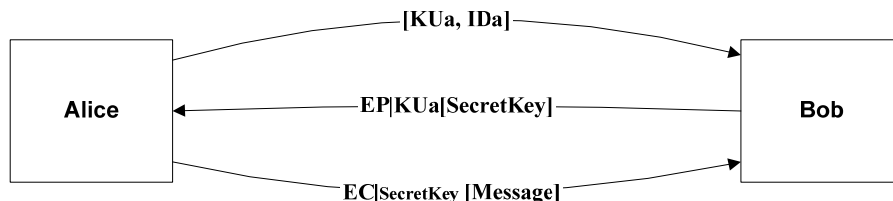


Figure 5: A Simple Scheme to Distribute Secret Key (Merkle, 1979)

At the end of the 1980s Kerberos Version 4, which is a network authentication service developed by Project Athena at MIT, came into being. Later Kerberos Version 5 corrected some drawbacks of Version 4. Kerberos provides a pretty successful centralized mutual authentication service between servers and users, but it is very complex system and is vulnerable to a password attack (Steiner, Neuman, & Schiller, 1988). The prototype protocol this paper suggested makes use of some similar concepts from Kerberos, enhances password protection and avoids the complexity. Based on Secure Shell Protocol, many secure products have been developed to secure the internet communication. The Secure Shell Protocol Version 2 is standardized by the Internet

Engineering Task Force (IETF). But the Secure Shell Protocol has two drawbacks. The first one is that each client keeps its own private key. Each user need to run ssh-keygen function to generate its key pair, which stores the private key in \$HOME/.ssh/identity and the public in the \$HOME/.ssh/identity.pub in the user's home directory. The user should then copy the identity.pub to \$HOME/.ssh.authorized\_key in his/her home directory on the remote machine (Openbsd). It is very cumbersome and insecure for the user to generate his/her own key and perform the copy. Nevertheless in the protocol this paper suggested the keys are all managed by the server side. The second drawback is that the password needs to be sent to the server via the network. It is possible for the crackers to steal the password by whatever means. In the protocol this paper suggested, the password is only used locally by the client and the server. The password does not go to the network at all. So the crackers have no way to capture the password from the network.

The security of a protocol highly relays on the Encryption and Decryption algorithms, the protocol adopts. This paper talks about a prototype protocol not the algorithm, so the focus of discussion is on the protocol. Of course when all the algorithms the protocol applies are sophisticated ones, the protocol will be a very secure protocol. Performance is a very important issue for protocols, for it is the base of communications. That is the reason why people always make and love fast machines. If a protocol is 100% secure, and performance is 100% slow, that is a scenario for not using the computer at all. The text that follows provides a comparison of several secure virtual terminal services to the method used herein.

## **Kerberos**

Both protocols provide authentication functions and need a database to store user information in order to authenticate the sender and the receiver.

However, Kerberos authenticates the user by way of tickets, small sequences of bytes with limited life times, while the user's password remains secure on a central machine. My protocol use password based encryption to authenticate the client and the server. The server connects to the secure database containing all users' account information. Other differences are listed as follows.

1. Compared with the protocol this paper suggested, Kerberos is a very high overhead system, as far as the setup is concerned. It uses authentication server (AS), Ticket Granting Server (TGS), and a number of application servers. Significant host infrastructure is required before it can be used. My protocol is designed to minimize overhead and so therefore is easily deployed and designed to work on existing system with minor changes.
2. A Kerberos server must share a key with each host server. In my protocol the server only needs to share a key with the clients.
3. Kerberos only provide authentication service. Whereas, my protocol provides mutual authentication and a secure channel between the client and server.

## **PGP (Pretty Good Privacy)**

PGP is a popular encryption program available free worldwide in versions that run on various platform. It can authenticate users and encrypt data files and email messages. These functions are similar to my protocol. The differences are listed as follows:

- 1.PGP uses the combination of SHA-1 and RSA to format a digital signature to authenticate the sender. DSS/SHA-1 can be the alternative scheme to format the digital signature. My protocol use password based encryption to provide mutual authentication between the client and server.
- 2.PGP implements confidentiality by using CAST-128, IDEA or 3DES. My protocol can be configured to use double encryption to implement the confidentiality. The double encryption can be the combination of two different symmetric encryption algorithms or use one encryption algorithm twice by applying two different keys.
- 3.PGP is file-based typically encrypting one file or email message at a time on a single computer. My protocol encrypts an ongoing session between the client and server.

## **SSH (Secure Shell)**

SSH provides authentication and confidentiality for the client and server communication. These functions are similar to mine. The differences are listed as follows:

- 1.In SSH, the password needs to go through the network via a secure channel. The encrypted password may be captured and analyzed by crackers. In my protocol the password does not go through the network at all.
- 2.SSH is a very sophisticated protocol, it guarantees privacy of your data, integrity of communication, mutual authentication, authorization and forwarding or tunneling to encrypt other TCP/IP-based sessions. SSH may select from among the following: Kerberos, Rhosts, RhostsRSA, Public-key, TIS or Password to authenticate the sender and receiver.(Barret, 2001) Depending on how it is configured it can be very resource intensive, whereas my protocol is designed to minimize resource intensity. Currently, my prototype protocol provides mutual authentication and confidentiality, but additional functions could be added to my protocol to meet additional security needs.

## **METHODOLOGY**

### **Introduction**

In the implementation of this protocol, the server is assumed to be a trusted server and it has access to a secure database, which keeps a record with the information of each legitimate user and his/her authorized services. The security database keeps information such as, User I.D., password, Public Key (KU), Private Key (KA), etc. Depending on the encryption and decryption algorithm that is chosen, the fields in the record of the database may vary accordingly. Both the server side and the client side are equipped with a common secret key (csk), which can be either hard coded in the source code or additional secret key exchanging process can be added so as to

update the secret key properly. Both the client and server sides must be equipped with the appropriate symmetric and asymmetric encryption and decryption functions. Given these prerequisites, the client and server can accomplish mutual authentication and confidentiality in communication by following the protocol phases.

### **Establishing the Secure Communication Channel**

1. Set up the secure network connection between client and server after a user launches the client application and punches in his/her user I.D. and password.
2. Use the common secret key known to both sides and a symmetric encryption and decryption algorithm to safeguard the input stream from the network and output stream to the network.

### **Mutual Authentication**

1. Via the established secure communication channel in the previous phase, client sends the user I.D. only to server.
2. Server checks the user I.D. against the maintained database to find the record of that user. If there is a match, this user I.D. is valid; otherwise it is not.
3. When the user I.D. is valid, the server uses the password and a predefined function to derive a new key and uses the new key to encrypt the user I.D. and send this information back to client via the secure channel.
4. On receiving the message sent by the server in step 3, client tries decrypting the message using a key, which is derived for the user-provided password by using the same predefined function, which the server uses to derive its new key. If the recovered message is the user's I.D. then the message is properly recovered and the use-provided password is correct. The client and server also authenticate each other.

### **Enhancing the Secure Communication Channel**

1. After the mutual authentication, the server sends the client's private key over the secure channel.
2. After the client has got its private key, the server will use the client's public key to encrypt a session key and pass the message to client.
3. The client will recover the session key from server by decrypting the message using its private key.
4. After the session key exchange, the client and the server proceed to use the session key via a symmetric encryption algorithm to encrypt outgoing message and send the encrypted message over the established secure communication channel. At the other end, the message is recovered by the same session key and symmetric decryption algorithm. Actually, the plain message is encrypted twice. One by the session key in conjunction with the symmetric encryption algorithm in this step; the other by the secure communication channel set up at the

very beginning. In this way the secure communication channel is enhanced. The diagram of the phases is depicted Figure 6.

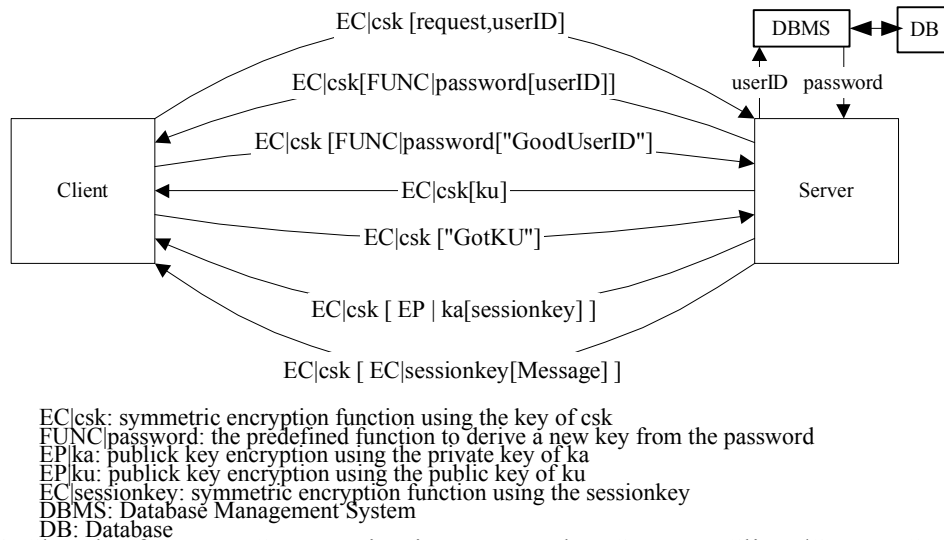


Figure 6: Sketch of Secure Communication Protocol to Support Client/Server Computing

## CONCLUSION

This paper presents a prototype protocol, which applies symmetric and asymmetric encryption techniques to safeguard the client and server network computing. This protocol provides the client and the server with mutual authentication, secure communication channel and it also solves the most important security problem on the Internet, hackers stealing passwords by using the password locally only. The following future work can be done to make the protocol more robust, flexible.

A login failure time limit mechanism can be added, that is, when a user fails to login onto the server for a limited times, the server precludes the connection from that IP address for a certain period. This can prolong the login time when a potential opponent tries logging in by user I.D. guessing or password guessing. And also the system can take measures to make illusion for the attackers, that is, when someone provides a wrong user I.D. or password to login onto the server, the server would not stop the session immediately; instead, it should start the logging process and inform the system administrator of that. By this means, the server can waste the time of an intruder trying the password-guessing attack and make the password attack harder than ever. Meanwhile, an encryption algorithm negotiation mechanism can also be added to the system. More encryption algorithm functions can be equipped to server side and client side. After connection, client and server make a negotiation about which symmetric and asymmetric algorithms are to be used and then start the communication. This will make the hacker's work even harder.

## REFERENCES

- Barret, D. (2001). *SSH, the secure shell: The definitive guide*. CA: O'Reilly.  
 Chadwick, J.(1990).*The decipherment of Linear B*. New York: Cambridge University Press  
 Cormen, T. H. (1997). *Introduction to algorithms*. Boston: McGraw-Hill.  
 Kahn, D. (1996). *The codebreakers: The story of secret writing*. New York: Scribner

- Merkle, R. (1979). *Secrecy, authentication, and public key systems*. Ph.D. Thesis, Stanford University.
- Openbsd.org. Retrieved September 15, 2006 from <http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1>
- Rivest, R., Shamir, A., and Adleman, L.(1978) A Method for Obtaining Digital Signatures and Public Key Cryptosystems.” *Communications of the ACM*, February 1978
- Stallings, W. (2006). *Cryptography and network security: Principles and Practices, 4<sup>th</sup> Edition*. New Jersey: Prentice Hall.
- Steiner, J. G., Neuman, B. C., & Schiller, J. I.(1988). Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the Winter 1988 Usenix Conference*. (Version 4). Retrieved from <http://web.mit.edu/kerberos/www/papers.html - krb4>
- The Internet Engineering Task Force. Retrieved September 15, 2006 from <http://www.ietf.org>