Designing Optimized and Secure File Transfer Application

Daniel Emerie
HiLCoE, Software Engineering Programme, Ethiopia
daniemerie@gmail.com

Mehari Misgana
HiLCoE, Ethiopia
m.g.msgna@gmail.com

Abstract

There is a considerable increase in the exchange of data over the Internet and other media. This data may contain confidential information that needs to be secured from any third-party access. To do so encryption plays an important role to ensure confidentiality and integrity. The use of different encryption algorithms has different advantages and disadvantages in terms of computational time, resource utilization, and security levels. Security and performance have an inverse relationship to each other, which increases security level and performance should be compromised and vice versa.

This has motivated us to design a file transfer system that can compromise the trade-off between security and performance. It improves performance by compromising security and increases security level by compromising efficiency. This is managed using an auto system. The implemented auto system optimized the system from 3.8% up to 39.4% performance enhancement depending on the size and chosen encryption algorithm.

The file transfer application uses RSA algorithm for secure secession key exchange. It uses RC4, Blowfish, and AES algorithms for file or data encryption. Open SSL digital signature is used for authentication, which improves the security level of file transfer. Both sender and receiver authenticate each other using digital signature. The application is also platform independent.

Keywords: Advanced Encryption Standard; Optimization; Authentication; Self-signed Certificate

1. Introduction

Simply having the capacity to transfer a file from one location to another is not enough. Businesses today face multiple security threats in highly competitive environments. They need a secure file transferring system to securely transfer their sensitive business-critical data. Governmental and non-governmental organizations also need a secure file transfer system to share highly confidential and sensitive information. Globalscape defines sensitive information as information that is protected against unwanted disclosure [5].

History of information security begins with the history of computer security after Robert Morris created what is now widely acknowledged as the first computer worm in 1989 [6]. The use of computers has been concentrated on computer centers, where the implementation of computer security focuses on securing physical computing infrastructure. Although the openness of Internet-enabled businesses helps to quickly adapt its technology ecosystem, it also proved to be a great weakness from an information security perspective. New security threats are emerging every day from malware that can be inadvertently installed on a user‘s machine [7]. So, information security is becoming crucial to all organizations to protect their information and conduct their business.

Security is a concern to IT and finance departments. This concern about security is often spread pretty thin. Because of this, an IT department can make itself very unpopular when it insists on solutions and procedures that slow things down or impede people as they go about their daily jobs. This paper is intended to examine the impact of security measure overhead against performance and design a secure and optimized file transfer application [4].

2. Related Work

Since the early days of the Internet, file sharing is one of the major tasks of organizations. It is very sensitive and requires strong confidentiality, integrity, and proper authentication to share files [4]. On the other hand, there are some organizations that can’t tolerate delays.

Over the past few decades, cyber security has gained crucial importance in the way businesses operate and survive in their value systems. Information security is possibly one of the most vibrant areas in the IT sector, in which technical innovation constantly paves the way to defeat emerging threats. This is not surprising, as the threat landscape itself is constantly evolving and it demands a constant revival of defense tactics [2].

According to ipswitch.com, the use of FTP servers to transfer files has been popular for many years, and the FTP protocol was once considered the easiest way to move business data. Studies estimate that it is used by approximately 80% of all businesses. Compliance audit firms often see these environments as a red flag and the U.S. FBI even issued an industry alert bulletin cautioning businesses about the risk that FTP servers can present [9].

Security is playing a vital role in the field of communication systems and the Internet. According to iplocation.net, AES, RSA and Blowfish algorithms are among the popular encryption algorithms in 2018 [10]. This paper presents the overhead of each algorithm, i.e., RC4, Blowfish, AES and RSA and designed an optimum file transfer application by decreasing performance overhead.

2.1 Security versus Performance

A data encryption algorithm would not be of much use if it is secure enough but slow in performance because it is a common practice to embed encryption algorithms in other applications such as e-commerce, banking, and online transaction processing applications. Embedding of encryption algorithms in other applications also precludes a hardware implementation, and is thus a major cause of degraded overall performance of a system [13].

According to Rusen, the impact of encryption on general computing performance, PCMark Vantage runs several tests and its average score is 7 testing areas: Memories (working with photos), TV and Movies, Gaming, Music, Communications, Productivity, and HDD. The scores were highly impacted by encryption. The tests revealed interesting results. It seems that encrypting a computer with True Crypt will have a reasonable impact on the performance of a system [1].

There are many approaches to support implementation: client-server, peer-to-peer, broadcast, and multicast. But according to Dorin Custură, in client-server architecture (like FTP or HTTP), each client connects to the same server and retrieves data from it. Because of that, the bandwidth of the server's network infrastructure has to be large enough to cope with the increased output caused by each client connection. This may compromise the performance of the system [11]. This paper uses client-server paradigm to implement the file transfer protocol. But it doesn't have central repository or database; the communication takes place just between sender and receiver.

Therefore, we have conducted experiments to know the performance trade-off between security and performance. Having seen the result, we decided to do a performance analysis of encryption, and examine the performance of different encryption algorithms and finally the result is used to add auto system to choose different encryption algorithms for the optimization of performance.

2.2 Digital Signature

A digital signature serves the purpose of ensuring data authenticity and integrity. A digital signature is a technology based on applied cryptography using specific hardware and software tools. The digital signature creates a unique electronic record in the document which can be later re-verified to ensure no change was made to the document over time [8].

3. The Proposed Solution

The proposed FTP Application is a secure file transfer and message exchange application in a client-server paradigm. With an intention to address the limitations of the existing file transfer applications and protocols, the proposed FTP Application is designed to address *security*, *reliability*, *interactivity*, *interoperability*, and *efficiency*.

3.1 Security

The proposed FTP Application addresses two of the main IT security issues: authentication and confidentiality.

a. Authentication

Before uploading as a client, it should authenticate the identity of the file server so it won't leak the data to random entities including criminals. The server also authenticates the client before it accepts the file sent from client, using its digital signature.

To upload a file or send a message, the client has to contact the server at some advertised IP address. However, we cannot simply trust the IP address because the IP address can be easily spoofed.

It is conceptually simple to design an authentication application using public key cryptography. What a client can do is ask the server to sign a message using its private key and send that message to a client. The client can then use the server's public key to verify the signed message. If the check goes through, then since only the server knows its private key, but no one else, the client knows that the message must have been created by the server.

However, there is one catch here. How can a client obtain the server's public key trust? This problem is often solved by using digital certificates. Digital certificates can be issued by a trusted certificate authority (CA). In this case, the server can get its certificate signed by the CA. This certificate contains the public key of the server [12]. Therefore, the authentication mechanism of the proposed FTP Application is implemented using public key cryptographic with digital certificates. The server also needs to authenticate the client with the same way. An overview of the authentication process is illustrated in Figure 1.

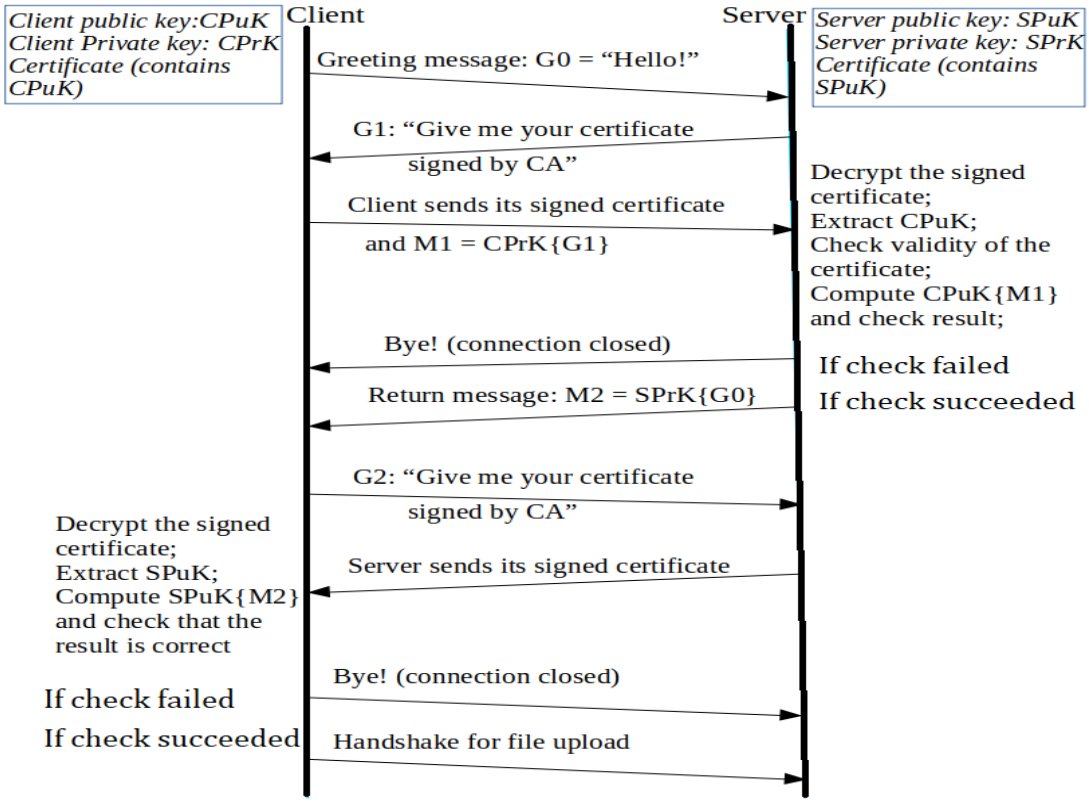

Figure 1: The Authentication Procedure

b. Confidentiality

While carrying out the upload, it should be able to protect the confidentiality of the data against eavesdropping by adversaries.

The confidentiality application can be implemented using either private key or public-key cryptography. Obviously, the latter has stronger security guarantee, but the performance overhead is also higher. For example, encrypting and decrypting large files using public-key cryptography could create huge latency, which might not be acceptable in some delay-sensitive systems.

Therefore, our plan is to implement the confidentiality application using both private and

public key technologies with varying key sizes. Therefore, it uses RC4, Blowfish, and AES from the private key and RSA from the public key cryptography. Then, we did an analysis to find out the better combination for better efficiency [3]. The decision to use either private key or public key could be decided manually (via users' choice) or automatically (the system decides at runtime). The AES encryption and decryption procedure is shown in Figure 2.

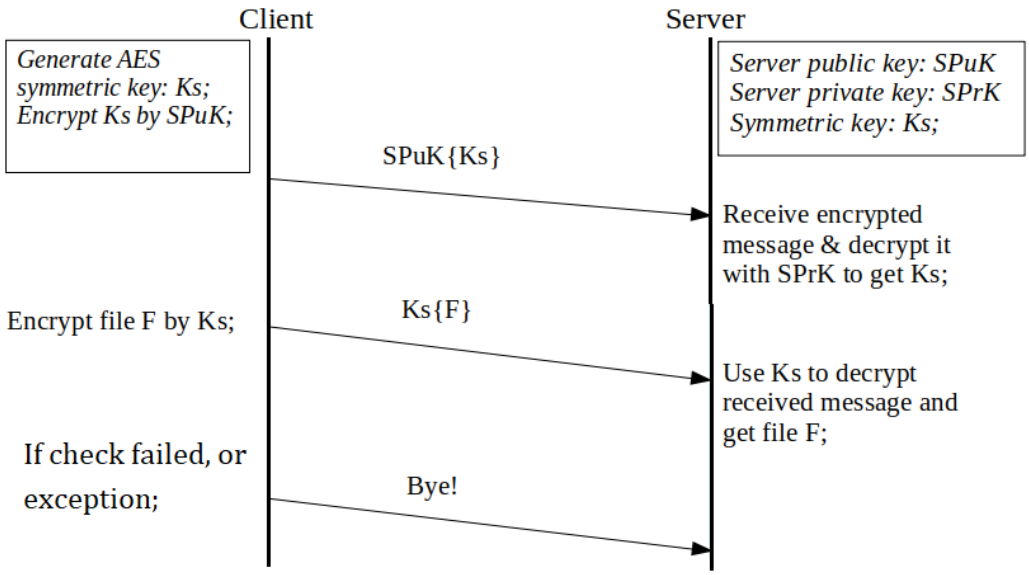

Figure 2: The AES Encryption and Decryption Procedure

3.2 Efficiency

The trade off between security and efficiency is a major concern. Due to its high computational cost, public key cryptography is limited only for key exchange. Thus, we intended to do a wide range of experiments over various performance factors and an optimal combination of the factors could be chosen manually or atomically.

3.3 Interactivity

One of the many features to be included to the proposed FTP Application is interactivity. Therefore, the client application is implemented with an interactive GUI. A preliminary prototype of the client application is illustrated in Figure 3.

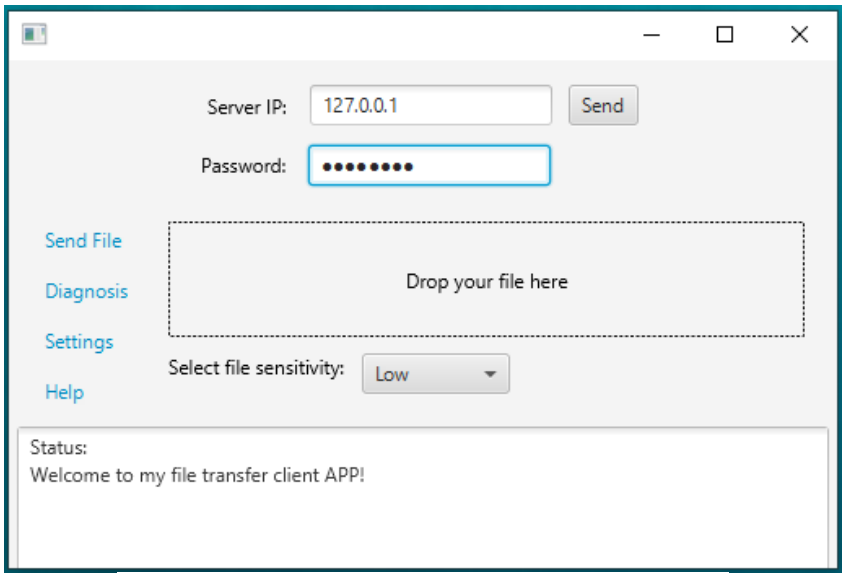

Figure 3: A Simple Prototype of the Client Application

Since the server only listens and accepts client requests, it is not necessary to add a GUI feature for the server application. It can be implemented as a command-line application. Then, it can simply run on command-line and listen to client requests. Upon successful authentication, the client can connect with the server and the file can be uploaded, but when receiving a file, the server has popup dialog box to notify the received file with open option.

3.4 Interoperability

The Java runtime environment is platform-independent. The FTP Application is implemented in Java and it will be compatible with several operating systems. It is tested on Windows, Linux, and Mac operating systems.

3.5 Reliability

Since the proposed FTP Application uses encryption and digital signature, the sender can establish secure session and can exchange encrypted data to ensure security while in transition.

4. Simulation Results

Performance measurement criterion is the time it takes by the algorithms to perform the encryption and decryption of the input text files. The following are the parameters that calculate the performance of algorithms.

Encryption Decryption Computation Time: Encryption time was taken from current time, and converted into common date and time format using SimpleDateFormat and Calendar method. Decryption took place on the server-side. Decryption started right after the cipher array was received. It ended after all received blocks or stream of files were completely copied as decrypted file.

Throughput: Throughput refers to how much data can be transferred from one location to another in a given amount of time. In our context, we used throughput to represent the amount of data encrypted per given time (kilobytes per 1 millisecond).

Overall elapsed time: This is the time it takes to process from the beginning of encryption in the client side till the end of decryption in the server side.

Performance overhead: is the performance overhead of one encryption algorithm over the other. This is used to acquire information about the overhead of one algorithm over the other. This information is used as an important input to design optimum file transfer application.

5. Experimental Results and Analysis

The experiment result is collected in a table by representing different sizes of files and corresponding encryption execution time taken by RC4, Blowfish, AES and RSA algorithms in milliseconds. Based on the information inferred from the result, we examined the outcome of the experiment-based parameters set between RC4, Blowfish, and AES.

The result of encryption shows the superiority of RC4 over other algorithms. It is the fastest algorithm, 1.6 milliseconds compared to 5.2 and 7.2 milliseconds for Blowfish and AES, respectively. The decryption also shows the same result with encryption where RC4 has superiority over both Blowfish and AES. But relatively compared with encryption, the difference in decryption is not as such remarkable.

Based on the experimental result, RC4 has an advantage over other algorithms in terms of throughput. The results show that RC4 has a very good performance compared to other algorithms. In addition, it shows that Blowfish has a better performance than AES, 135.7kb/s and 63.9 kb/s respectively. Amazingly, it clearly shows that Blowfish has almost 1/2 throughput of AES, or in other words, it needs double time than AES to process the same amount of data.

The test proved that RC4 is the fastest algorithm in terms of encryption and decryption time. On the other hand, AES is slower than both RC4 and Blowfish.

6. The Proposed “Auto” Selection Scheme

To enforce a scheme that automatically balances security performance trade off, Auto option is included in the FTP Application selection setting. This means, if the user chooses the Auto option, the file transfer application will automatically select the appropriate cryptographic algorithm (i.e., RC4, Blowfish or AES) so as to balance the security-performance trade off in transferring the file.

The decision was made by considering two main factors: file size and file sensitivity (i.e., security sensitivity of the file). As such, the scheme dynamically chooses RC4, Blowfish or AES at runtime in light of addressing neither the security nor the performance is significantly affected. However, it is well understood that it is not possible to address both security and performance at the same time since they are strong tradeoffs one over the other. As such, the proposed Auto scheme tries to improve the trade off by not significantly affecting either of them.

To make a decision in selecting the appropriate cryptographic algorithm, the Auto‖ selection scheme uses a 3x3 dimension matrix (as shown in Table 1) over the file size and file sensitivity categories. The file size is categorized into Small, Medium and Large. The file sensitivity is also categorized as Low, Medium and High.

Table 1: Auto Selection Scheme Matrix

	File Sensitivity		
File size	Low	Medium	High
Small	SL	SM	SH
Medium	ML	MM	MH
Large	LS	LM	LH

The file size is categorized based on the execution time of AES. For this, the following assumptions are considered. If AES takes around 2 seconds, the file is categorized as Small. If it takes more than 2 seconds but less than 10 seconds, it will be in the Medium size category. If it takes beyond 10 seconds, it will go to the large file category.

According to the above assumption, a file size below 30 MB is in Small; a file size between 30 MB and 100 MB is Medium and a file size beyond 100 MB is in large file size category.

The Auto selection scheme matrix with selected cryptographic algorithms is illustrated in Table 2.

Table 2: Auto Scheme Matrix with Selected Algorithms

	File sensitivity		
File size	High	Medium	Low
Small	AES	Blowfish	Blowfish
Medium	AES	Blowfish	RC4
Large	Blowfish	RC4	RC4

7. Conclusion and Future Works

This paper presented a performance evaluation of selected symmetric and asymmetric encryption algorithms. The selected algorithms are RC4, Blowfish, AES, and RSA.

According to the experiments, RC4 is moderately secure. Its key scheduling algorithm is somewhat weak, but it may be tolerable for low sensitive files. RC4 is the fastest encryption algorithm considered in this paper. As a stream cipher, it is well suited to the near-continuous flow of wireless and wired TCP/IP transmission. By contrast, the slowness of RSA due to the high complexity of modular exponentiation is not usually acceptable for encryption of large files. It is instead often used for small data parcels, such as signatures and keys.

Blowfish and AES have no security flaws to date, and as we see from the experiment, even if they are slower than RC4, they still have superiority in terms of security, and more enhanced performance than RSA, which, is why we choose Blowfish and AES for data encryption, but RSA only for key exchange.

RSA is not recommended for file encryption because its speed and computational cost are high, especially for medium and large files.

Future work will focus on improvements to the FTP Application of enterprise levels and by further experiments on security and usability issues. Including integrity check and network status evaluation for auto system will be considered in the future.

References

[1] Ciprian Adrian Rusen, https://www.digitalcitizen.life/what-performance-impact-system-encryption-truecrypt.

[2] Cybintsolutions, https://www.cybintsolutions.com/cyber-security-facts-stats/

[3] Prerna Mahajan and Abhishek Sachdeva, "A Study of Encryption Algorithms AES, DES and RSA for Security", Global Journals Inc., USA, 2013.

[4] Ethan Miller, *et al*., "Strong Security for Distributed File Systems", University of California, IPCCC 2001, April 2001.

[5] Globalscape, https://www.globalscape.com/solutions/secure-file-transfer

[6] Info Security Magazine, "Defining Moments in the History of Cyber-Security and the Rise of Incident Response", https://www.infosecurity-magazine.com/opinions/the-history-of-cybersecurity/

[7] Security and Privacy on the Encrypted Network, Version 1.5 Released: January 20, 2015, https://www.symantec.com/content/dam/symantec/docs/white-papers/securosis-security-and-privacy-on-the-encrypted-network.pdf

[8] SSL.COM, http://info.ssl.com/article.aspx?id=10241

[9] Ipswitch.com, https://www.ipswitch.com/resources/best-practices/ftp-software.

[10] Iplocation.net, "What are the most secure encryption algorithms", www.iplocation.net/encryption, December, 2018.

[11] Dorin Custură, Department of Economic Informatics and Cybernetics, Bucharest University of Economic Studies, Vol. IV, No. 1, 2012, Romania.

[12] Reshma Afshar, "Digital Certificate", Indiana State University, October 2015.

[13] Aamer Nadeem and M. Younus Javed, "A Performance Comparison of Data Encryption Algorithms", National University of Sciences and Technology, myjavedgceme.edu.pk.