Int. J. Advance Soft Compu. Appl, Vol. 6, No. 1, March 2014
ISSN 2074-8523; Copyright © SCRG Publication, 2014

Study on the Effectiveness of Securing Instant Messaging Using One Time Pad

Anthony Arifin, and Hazinah Kutty Mammi

Department of Computer Science, Faculty of Computing
Universiti Teknologi Malaysia
81310, Skudai, Johor
Malaysia
e-mail: thonyarifin@gmail.com, hazinah@utm.my (corresponding author)

Abstract

Instant Messaging (IM) is one of the popular applications for real-time communication. Along with its popularity the threats that target instant messaging also rising. Some studies have been conducted to improve the message confidentiality of the instant messaging, which do not possess by most existing IM. The purpose of this study is tried to verify the effectiveness of applying One Time Pad (OTP) encryption in the instant messaging system. This scheme is to address the weakness IM of transferring message in plaintext. Experiment was done on the proposed system to find out the performance. Results were compared against same IM system without encryption and also against similar system Results showed that there is probability to equip IM system with OTP encryption to secure the message transmission without decrease its performance.

Keywords: *Instant Messaging, One Time Pad, encryption*

1 Introduction

Instant messaging (IM) is an application where two or more users can communicate commonly via text message. Basically email and IM have the same concept, to deliver message to the recipient. But one thing that made instant messaging more popular is the ability to send message in real-time only in few easy steps.

However just like any other online application, the security issues of the information and personal data need to be taken into account. It's important especially if the information is sensitive or related to one's personal information. The safety of the transmission needs to be analyzed to find out whether encryption

can be used in order to maintain the information secrecy within IM communication.

Most of the IM application transfer data directly from sender to recipient without any protection to maintain transfer speed [1]. Such condition could be harmful if the information transferred through IM is sensitive. This condition can be exploited by adversaries.

With regards to this there is a need to protect data transferred through IM. By encrypting the transferred data, which in this study using One Time Pad (OTP) cipher, the secrecy of information can be preserved. Unfortunately by encrypting the data there is some perception that if it was done the transmission wouldn't be as 'real time' as before. Results of this study will help to decide whether it is possible and appropriate to equip an IM with encryption especially OTP without affecting the performance

2 Instant Messaging

Basic instant messaging functionality can be derived from the almost ancient UNIX "finger" and "talk" application – the ability to identify users online and to exchange small text messages [2].

Unfortunately there is a downside of IM that most of the users, even some administrators do not know. Most of the popular IM applications are 'port-agile', if their native port is closed they are capable to locating other open ports and tunneling their traffic over a different port instead [1]. This is where the security of the organization will be vulnerable. All request denied by the firewall will be able to follow this IM which is "free from firewall" to get in the organization system.

Additionally most of the IM applications send the message from sender to receiver in plaintext [2]. This could be the target of the attacker who wants to intercept or modify the messages that being sent.

2.1 IM System Architecture

Based on RFC 2278 released in 2000, a standard model of IM system has been defined. The model describes two services that must be running in an IM system in order to work properly: a presence service and an IM service. As stated in [3], the presence service serves to accept, store, and distribute the information, while the instant message service serves to accept and deliver instant messages to instant inboxes.

2.1.1 Presence Service

The presence service acts as the information distributor in the IM system. The information exchanged is the status of the user, the 'presence' whether they are online or not.

It consists of two components, Presentities and the watcher, as illustrated in Fig. 1. Presentities is the component that provides update regarding presence information to the presence service. This presence information will be stored and distribute by presence service. In the other hand the watcher is the component responsible to receive presence information updates.

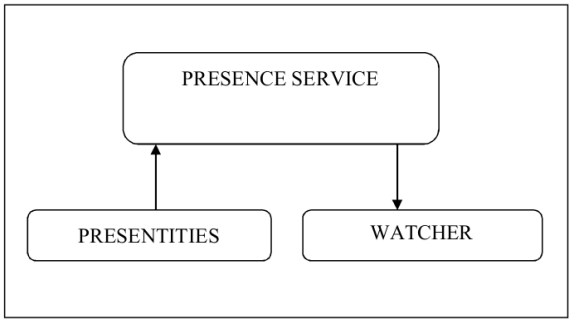

Fig. 1: Presence Service

2.1.2 IM Service

The IM service tasks are to receive and forward the message from sender to the recipient. It operates like a postman. It will take the message from the post box, bring it to the post office and later send it to the appropriate address. Within this service there are two components that act exactly like that, as shown in Fig. 2. The first one is the sender, which will send the message. The second one is the Instant Box, which will receive message. The IM service will be the post office, which will organize and forward the message.

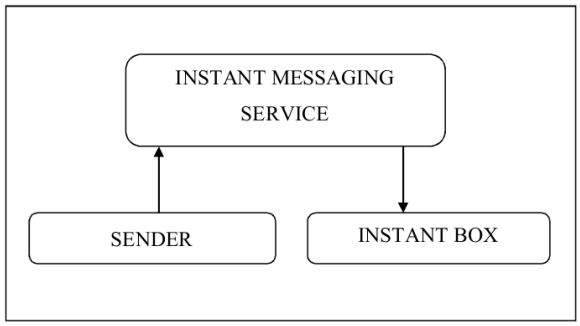

Fig. 2: Instant Messaging Service

2.2 Comparison of Popular IM Applications

Today there are many IM clients that can be downloaded and used on the internet. Due to the IM popularity, many developers raced to develop better IM applications. Table 1 shows some of the IM applications popularly used by most people and the comparison of the features provided by them.

Table 1: Comparison of Existing Instant Messaging Features

Features	Yahoo!	Windows Live	Skype	Google Talk
Chatting	√	√	√	√
Group Chat	√	√	√	√
File Transfer	√	√	√	√
Video Call	√	√	√	-
Voice Call	√	√	√	√
Photo Sharing	√	√	-	-
Encryption Feature	-	-	-	-

2.3 IM Security Threat

Since IM has become very popular these days, particularly for business, it is not surprising that it increasingly becomes target of attacks [4]. Some general threats that haunt the IM system since several years ago are as follows.

2.3.1 Plaintext Message Communication

As stated in [2] the communications done by most IM are done in plaintext message. This is the simplest yet very frightening problem for people who use IM to exchange sensitive information. This condition is susceptible to eavesdropping or sniffing attack.

Actually the main reason behind this behaviour is to make sure that the real-time characteristic of the IM maintained. The system must able to deliver the message in superb speed, as if the message is just written by someone next to the user. Therefore it is feared that encrypting the message will ruin this characteristic.

2.3.2 Denial of Services

This is the same attack that is used to strike the computer network system. The aggressor will send huge amount of crafted message to a computer or a server that is running the IM application. The target will face a very hectic time processing all received message which will cause the CPU and memory to hang or even crash, so it will not be able receive and process valid messages.

2.3.3 Phishing & Malicious Message

Another attack that has hit the IM application is fake messages. Since the system is basically an email, so the message can be sent directly from one person to another without adding that person into the contact list. Therefore an unknown person could send message to the user anytime.

3 One Time Pad Cipher

One Time Pad is originally invented by Frank Miller to protect the secrecy and privacy of the telegram transmission in 1882. However it was reinvented by Gilbert Vernam at the end of WW1. OTP is the simplest and most secure type of synchronous stream cipher [5].It was named one-time pad because the key usually written in a small pad and used only once. The main concepts of the OTP are:

1. The key only used once for each plaintext
2. Same key possessed by both party
3. The key must be truly random

One-time pad is theoretically the strongest algorithmic cipher [6]. It is based on the rules that the key must be truly random and it is not re-used. Therefore the adversaries can predict neither the key nor the plaintext.

OTP has property called .perfect secrecy which means the ciphertext doesn't give any additional information about the plaintext [7]. All this can only happen if the rules are followed properly. That is why the concept of destroying the key after use is the most crucial part of OTP. The security of the cipher text will be degraded if it encrypted with used key. Even the brightest cryptologist with super speed computer system won't able to break it forever.

3.1 Encryption and Decryption Process

There are two known variations of OTP that has been created and used today. All of them are totally secure if the concepts of OTP are applied properly.
The original OTP uses XOR operation. Every bit of the plaintext will be XOR-ed with the bit from key which produce the cipher text. In the Fig. 3 below the encryption and decryption process of the OTP using XOR operation is shown

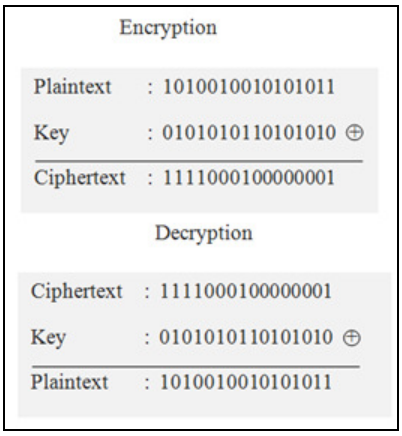

Fig. 3: OTP encryption and decryption process using XOR for bit

Another version is using modulo addition. Instead of change the data into bits, the encryption will process the letter using modulo 26 additive operations for changing the plaintext into ciphertext (encryption). Then to recover the plaintext it uses additive inverse. The overall process is similar to Vigenere Cipher, even the Vigenere table can be used as guideline to help encrypt and decrypt faster rather than calculate it manually. The numbering format for this type of OTP is started at 00 which is equal to A and so on until Z=25. Fig. 4 show the list of modulo 26 format for alphabet. In Fig. 5 the process of encrypting a text using modulo 26 is showed.

A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Fig. 4: Alphabet in modulo 26 format

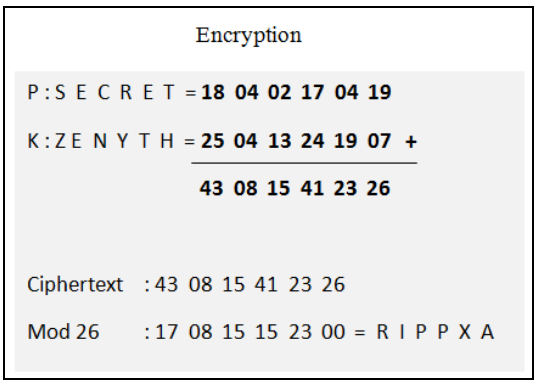

Fig. 5: OTP Encryption process using modulo 26

Despite its proven capability to achieve perfect secrecy, OTP has some drawbacks in practice. Those drawbacks are:

- The key must be as long as the message
- It is effective if the key is truly random
- There is difficulties to provide both party list of the key
- To produce randomness is not easy and expensive
- It will compromise the message if the key used more than once

.

4 Similar Studies

Security in IM transmission has been proved not save since the message are unencrypted and could easily sniffed by eavesdropper. Based on that circumstance several studies have been done to improve the security of IM. There is a study involving securing IM that have similar characteristic with this study except it using Elliptic-Curve cryptography. Other than that there is a plug-in called Pidgin Encryption which provides encryption services with concept Off-The Record (OTR).

4.1 Secure Instant Messaging and Presence Protocol (SIMPP)

In this study an IM system was designed and implemented a new IM application using Extensible Messaging and Presence Protocol based on Jabber standard. The server was using modified Jabberd which run in Linux and the client were developed using Borland C++ running in Windows. The application uses Elliptic-

Curve Cryptography (ECC) to maintain transmission confidentially and authentication communicating with each other. The message was encrypted using AES with CBC mode that used 128 bits key [2].

4.2 Pidgin Plug-in

Besides that study there is another IM system which provides encryption for public use, in other word for free. Mostly encrypted IM only can be used by enterprises which maintain its secret or information for insider only. As its name this plug-in only run for Pidgin. It is an IM application that released in 1998 with former name GAIM which not as popular as Yahoo Messenger for instance.

The encryption plug-in provides up to 4096 bit RSA encryption using the NSS crypto library from Mozilla. Keys are automatically transmitted and stored, making it very easy to use, but also resistant to man-in-the-middle attacks [9].

5 Design & Experimentation

In order to prevent IM system from send plaintext message in the transmission, an encryption process was applied on the IM process flow. The system was consisted of two parts, the server and the clients. A prototype of the proposed system was developed to evaluate the performance of the selected method to secure IM message transmission. Fig. 6 shows the architecture of the proposed IM system.

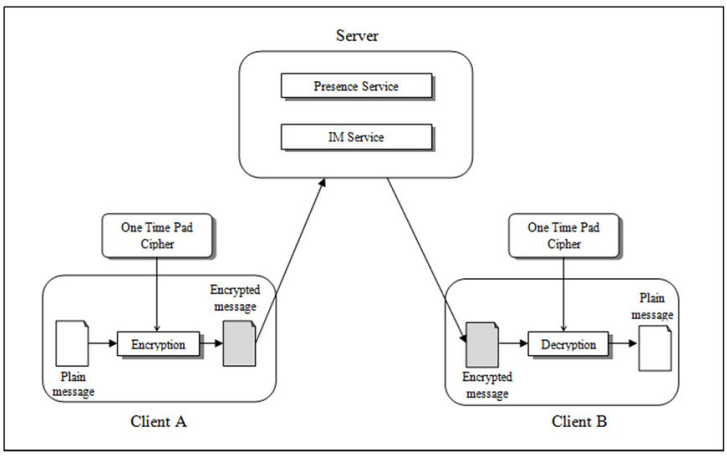

Fig. 6: Proposed IM System Architecture

As shown in the Fig. 6 the cryptography process is introduced before the message is sent and after the message is received. The clients communicate with each other through the server. Although so the server didn't know what messages being forwarded since they were ciphered. The server tasks only to identify clients and delivering the messages to them.

5.1 Experiment Design

The proposed IM system which developed as a prototype has several modules that run when the experiments were performed.

5.1.1 Server Module

The server acts as the middle-man tasked to record clients' connection and passing the message to all clients. After it started, server would listen to any incoming connection that came from the clients and process it. Server availability must be maintained since it is the heart of the system. Fig. 7 shows the steps taken by the server program when run.

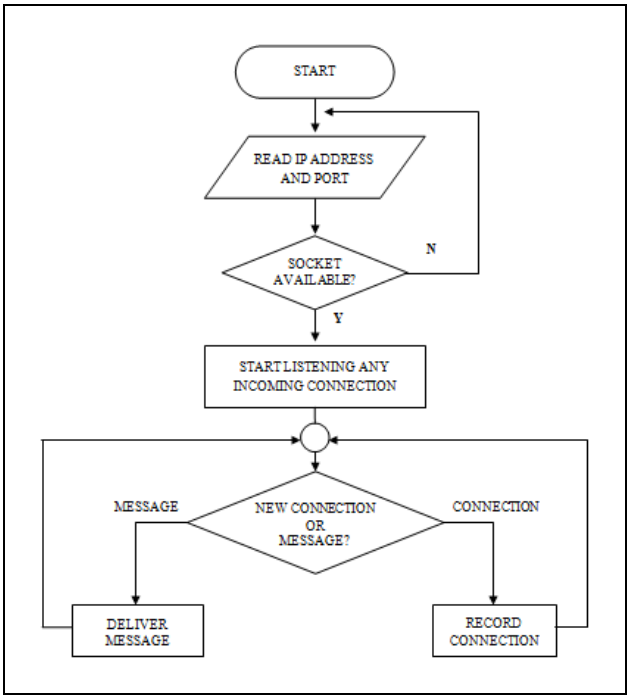

Fig. 7: Server Program Module

The server would kept waiting for any connection and processed it accordingly. If it was a message it delivered to the clients, and if it was a new connection initiated by a client then it recorded it and include in the communication.

5.1.2 Client Module

Instant messaging clients were interacting with each other by sending message that maintain by the server. They were the important aspect in this study. All that had been done were centred in the communication that performed by these clients of IM. Fig. 8 displays the flow of the developed IM client program.

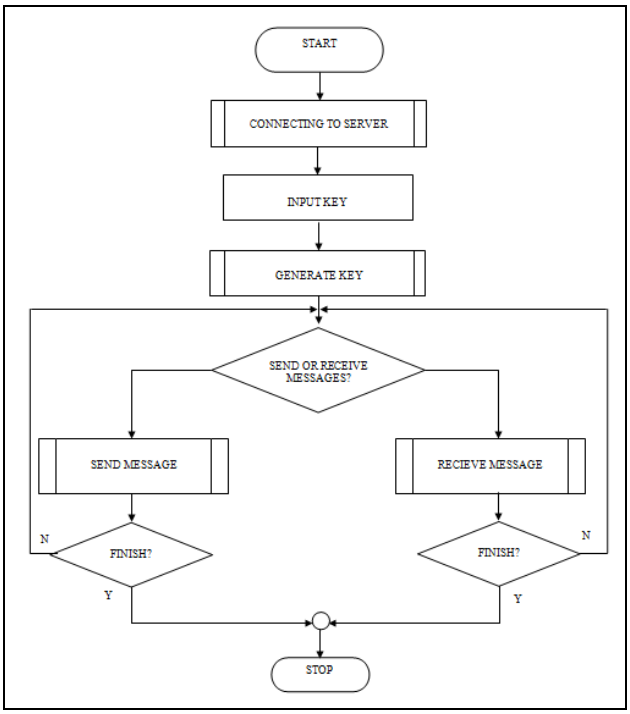

Fig. 8: Client Program Module

At the beginning the client was asked to input the username, IP address and also port number that used by server. Then the program tried to create connection with the server. If server with those details was existed then connection would be created. If not an error information would be given. Next, the server would verify the username. If it was existed in the server record then client had to re-enter new username.

After connection has been created, the client had to input a key which later processed to get a set of key that can be used more than once. The key generation procedure is going to explain later in this chapter. After key had been generated only then the message transmission can be executed.

5.1.3 Key Generation Module

This was the module that responsible to generate a set of key from key input by the client. The pseudo code for this module is as shown in Fig. 9 below.

```
START

    1. Read key input by user
    2. Get desired key size
    3. Remove non-alphabet characters, accept whitespace
    4. Repeat
            NewKeys[set_no] = key_substring[set_no] to
                                key_substring[set_no + desired key size – 1]
       Until set_no == (key_size– desired key size)
END
```

Fig. 9: Key generation Pseudo code

Based on that pseudo code, the number of new keys that produced was equal to key's characters length minus the desired key size. But since the set number was the offset of the array, which started by 0, then the number of set ought to add by 1.

For example if the key provided is "*computer*" and the desired key size is 4, then the number of set is:

$$(8\text{-}4) + 1 = 5 \text{ sets}$$

The new keys are *comp*, *ompu*, *mput*, *pute*, and *uter*.

This method was used because to generate truly random number was quite difficult and consumes time. If common word used a lot here, the risk for the message to be revealed became greater. Moreover it would be very tedious and annoying if new key had to input every time message needed to be sent. Therefore to minimize the risk and maintain the rules of OTP this method was applied. Just like OTP each key set only used once in each communication. That means after one message encrypted with a key set and sent, that key was removed and next message was encrypted using next key set.

5.1.4 Encryption and Decryption Module

This module was used to perform OTP encryption toward messages before it sent and when it arrived in the recipient computer. The encryption and decryption was using OTP modulo addition, which processed the text in modulo 27 (alphabet and whitespace). Fig. 10 shows the encryption and decryption process.

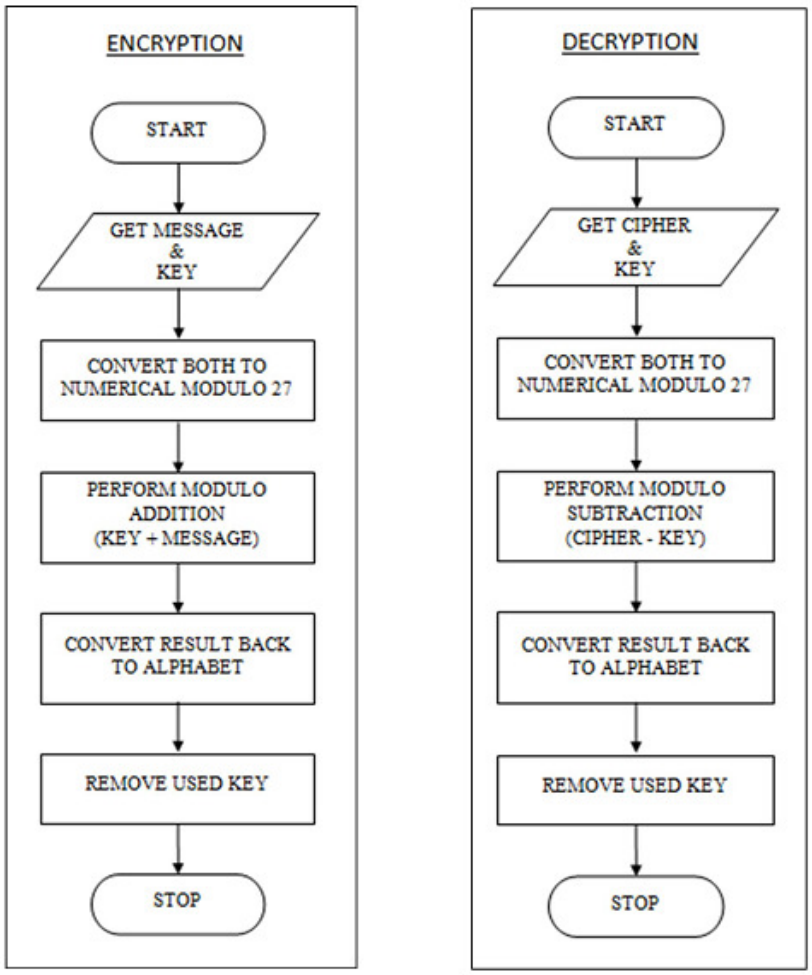

Fig. 10: Key generation Pseudo code

Both modules, as shown in Fig. 10, were performing nearly similar operation which were; getting input, converting, performing modulo operation, reverted it back to alphabet and removed used key. The difference was on the input and modulo process. Encryption module took plain message as the input, while decryption module took cipher message. Other dissimilarity was on modulo operation. Modulo addition was performed in the encryption, and on the contrary modulo subtraction was done in decryption.

On the second step the key and message were converted into modulo 27 numbers. This was done by convert each character into ASCII number. In order to do so, all characters were converted to capital letter and capital 'A' became start point. Hence 'A' = 65 in ASCII was equivalent as 'A' = 0 in modulo 27, and so forth.

In this proposed system, modulo 27 was consisted of alphabet and whitespace. The whitespace was added, after considering the readability of the transferred message. If whitespace were excluded, the message that sent would be hard to read. Since this system recognized only alphabet and whitespace, therefore the message could only consist of alphabet and spaces. Other characters wouldn't able to be decrypted correctly. Fig. 11 shows a table of the alphabet and whitespace number in modulo 27.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13
O	P	Q	R	S	T	U	V	W	X	Y	Z	sp	
14	15	16	17	18	19	20	21	22	23	24	25	26	

Fig. 11: Modulo 27 Table

Each key set was used to encrypt or decrypt a message regardless the different of total character. Consequently if the key set was not long enough for the message, the key would be repeated until it covers all messages.

5.1.5 Timer Module

The timer module was responsible to record the duration of vital processes in this system, which was required for the analysis phase. Timers were placed in both client and server. Fig. 12 shows parts that were timed.

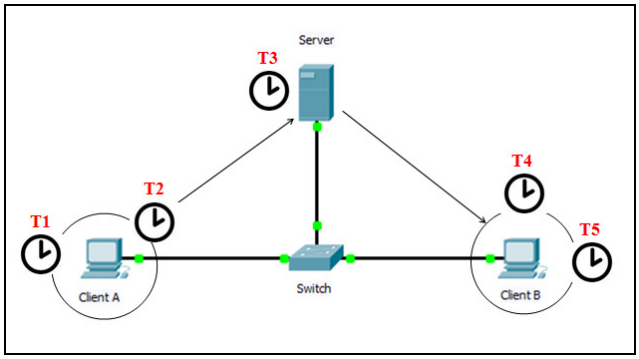

Fig. 12: Timer Position

Based on Fig. 12, processes that were timed are as follows:

- Duration of encryption process from the time user press enter/send button (T1)
- Current time when message is sent from client A to server (T2)
- Unnecessary time in Server (T3)
- Duration of decryption process (T4)
- Current time of decrypted message displayed in B (T5)

The database for the experiment was deployed in the server. All this data were transferred to server by the clients and then saved in to the database.

Several formulas have been considered to get the most accurate total time but most of them are tend to include unnecessary time in the process. Eventually the total time for each sending and receiving message was calculated based on this formula:

$$Total\ process\ time = T1 + (T5 - T2) - T3$$

Where:

$T1$ = Encryption time in A

$T2$ = Time client A start send message

$T3$ = unnecessary time in server

$T5$ = Time decrypted message displayed in B

This formula was providing more accuracy in calculation of total time compared to other because it deduct the unnecessary time. The unnecessary time in server was the time wasted to do timing and database related process for timing system, which had nothing to do with the IM system.

Duration to decrypt message (T4) wasn't counted in this formula because it was already included in the time to display. The purpose of recording this process time separately was to get precise duration of the process, which would help in analyzing the result.

Time taken in this system would be in the 100 nanosecond unit (ns) equals to 10-7 second. The reason was because most of the process that were done by the IM system finished under 1 second. Hence to able displaying process time with better accuracy this unit was chosen.

5.2 Experiment Setup

The experiments were performed in controlled environment, where 2 computers communicated using developed IM application with a server at the center acted as the mediator. Controlled environment imply that the experiments were done in closed set, there wasn't any actual users involved in the experiments. Besides that the controlled environment also specified that the network connection was in normal state without any disruption from outside.

All computers were connected to local area network via a switch using a straight through cable. All computers must synchronize the time using NetTime software to ensure the timer produce accurate result. Microsoft .Net Framework 4.0 must have been installed in all computers (server and clients) to ensure the application could run properly. The computers only run the application when the experiment was carried out.

The experiments were done by performing chatting activity between clients using prepared computers. Key size and message size were fixed before the experiments. Key sizes that were used are 3, 5, 10, 20, and 30 characters. Message sizes also varied from 5, 15, 30, 60, and 100 characters. A list of keys and messages had been prepared beforehand. Characters beside alphabet and whitespace weren't allowed to be used. Table 2 shows key and message that were used for testing.

Table 2: Pre-Defined Key and Message for Testing

Key	Key Size Generated
tsunami	3
waterfall	5
earthquake tornado	10
volcano sandstorm blizzards	20
landslide cyclone lightning flood geyser	30

Message		Characters Length
hello	M1	5
nice to meet you	M2	13
rainwater could contains pollutants	M3	32
elephants are large herbivorous mammals that live in the forest or bush	M4	60
a rainbow is an optical and meteorological phenomenon that is caused by reflection of light in water droplets in the atmosphere	M5	107

5.3 Parameter Testing and Method

As mention earlier in this chapter there were several parameters that involved in this experiment. One of the parameter was the key sizes. The process time depend on the key size. Therefore the selection of the proper key to be used in the encryption process was critical. Other parameter was the messages that were used in the experiment. Different sizes of messages were transferred to find out the system performance. Next thing that need attention is the time/clock in each computer. Since the timer recorded the local time in each computer and processed it, ensuring that all computers' time had synchronized was important.

Measured parameters weren't so different with involved parameters. First was processing time. Time taken to perform all tasks in these experiments was the significant factor that would be compared with other similar system. Next parameter was key size. Based on the result, suitable key size that showed effectiveness to entire process shall be determined. Lastly the strength of the encrypted message was tested and analyzed by performing brute force attack calculation and cryptanalysis on the cipher which was Kasiski Test.

5.3.1 Testing Method

The test was done by sending all 5 predefined messages using same key for 2 times alternately from each client. So, in total there were 4 set of conversations were done for one key size. Steps that were taken for the test were:

1. Start IM server application, NetTime program and MySQL server on the server. Start IM client application and NetTime program on the clients
2. IM clients established connection to server
3. Clients inserted the key and determine the key size, then new keys was generated
4. Synchronize the time of all clients' computers with server by updating the NetTime program
5. Start sending predefined set of messages alternately (client A, then B, then A again and so on)
6. When all messages had been sent, disconnect from IM server
7. Repeat from step 3 with same key and shift the initial sender to other client. If each client has sent first message twice (M1) continue step 8
8. Disconnect from IM server and Repeat from step 2 with new key

From those result, average was taken for each key size. By repeating the process for each key size couple times, data gathered would be more reliable and accurate. The clients were sending message alternately to emulate real chatting process.

6 Results, Analysis and Discussion

The result analyses were separated into three parts according to the measured parameters that were done in testing phase. The processing time taken when performing test is explained, followed by deciding appropriate key to be used and also the strength of the cipher produced by the proposed system.

6.1 Processing Time Analysis

After the test were done and result were gathered, some conclusion can be drawn directly by looked at the raw data. First criteria to be evaluated were the process time for each message. As expected the proposed system with cryptography function was working effectively, this was proven by the average process time of all data that shown on Table 3.

Table 3: Overall Average Process Time per message of Proposed System

Messages		Overall Average Process Time
hello	M1	0.0360616 s
nice to meet you	M2	0.0355152 s
rainwater could contains pollutants	M3	0.0394202 s
elephants are large herbivorous mammals that live in the forest or bush	M4	0.0317651 s
a rainbow is an optical and meteorological phenomenon that is caused by reflection of light in water droplets in the atmosphere	M5	0.0409846 s

Table 3 lists the performance of the system when being tested for each message. Overall the proposed system took time less than 0.05 second to process messages less than 100 characters. This duration was including encryption, sending and receiving, decryption and also displaying message time.

Since one of the aims of this study was to find out the effectiveness of using encryption in IM system, a comparison against similar system must be done. One of the ways to know it was by comparing the performance between developed IM system with encryption and developed IM system without encryption. Fig. 12

showed the comparison of the proposed IM system with encryption and proposed IM system without encryption.

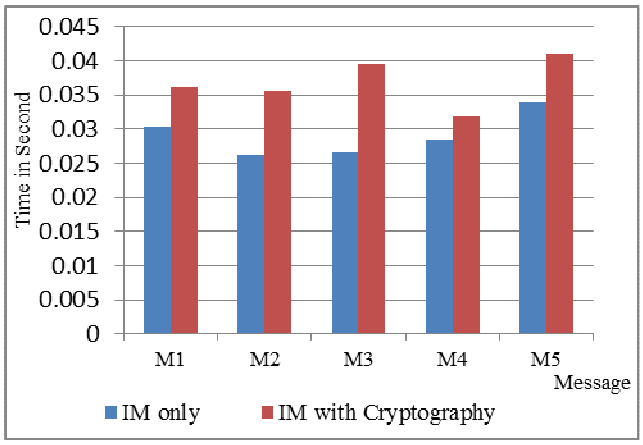

Fig. 12: Comparison of Process time between IM only and IM with cryptography

As shown in the figure, time differences between two systems are not that significant. Generally the times taken to process increased less than 0.015 seconds. Such increment in process time won't affect the overall performance of the system. Hence it can be assumed that it is appropriate and effective to use encryption in the IM system, which in this case One Time Pad (OTP) cipher.

6.2 Key Size Analysis

In this experimentation several key sizes had been used to check the performance and provided more data for the proposed system. The key sizes that had been user were 3, 5, 10, 20, and 30 characters. Fig. 13 shows average process time for each key size when encrypting predefined messages.

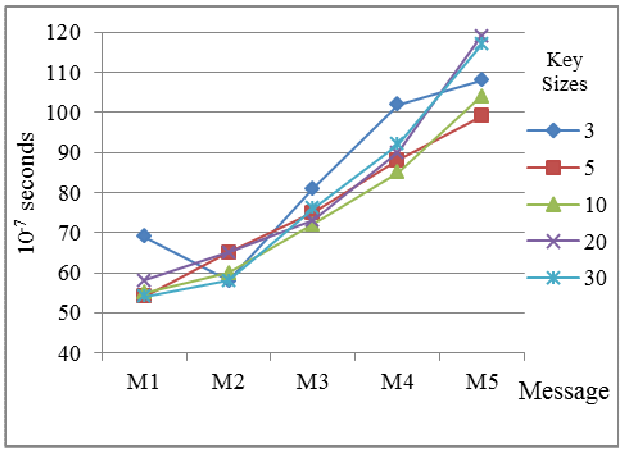

Fig. 13: Average Process Time for Encryption

Fig. 13 shows that time taken to process the messages were almost equal although the key size were different. The processing time increase as the number of messages increase. It didn't affected by the size of the key. Based on this analysis it can be conclude that any sizes of the key used to encrypt the message wouldn't much affect the overall performance of the computer, but the message sizes did have effect on the performance.

6.3 Ciphertext Strength Analysis

Ciphertext that produced during this test should be strong enough against any attack, since each message was encrypted using different key. This is the main drawback of using stream cipher which in this study was OTP. The availability of the key was crucial since a key cannot be used more than once. The key generator provided solution to overcome the problem with the key availability. The client can use a single key for multiple message transmission. It didn't re-use the key, instead it generate several keys from that single key.

In order to prove the strength of the ciphered message several tests have been done. The first one was calculating time taken to perform brute force against the ciphered message. The brute force attack tried to break the cipher by trying all possible characters available. Against this brute force attack the ciphered message was safe because time need to break the message would grow as the size of the message increase.

If the attacker was using all characters in keyboard which there were 95 characters including small and capital alphabet, numbers 0-9, symbols and the whitespace. The attack tried all combination of these characters to find the message. Details are as follows:

- To find 4 characters need : 954 = 81,450,625 possibilities
- To find 5 characters need : 955 = 7,737,809,375 possibilities
- To find 6 characters need : 956 = about 700 x 109 possibilities

The possibilities were increase exponentially as the size of the message growth. That was why brute force attack weren't a problem for a stream cipher.

Next test was trying to break the ciphered message using Kasiski test. Since the encryption process was applying Vigenere cipher in the process there was a possibility that the message could be exposed under Kasiski Test. Consider this ciphered message from one of the communication during the test:

TRKT GUFPS LSSGSFIL
WTCTTEXSDYLWHJFEGYBVSESGAXEHEWGGETLZULRBKRWTLLXVTUPT
JWZDWWL HFRHYREAYALRBFRPTKYJRXJFIDWMKRBFRM WTTKFGJI
WKX

That message was encrypted using 3 characters key. Using a Kasiski tool available in the internet, repeated substrings for this particular ciphered message was found.

Repeated Substrings:
LRB at index 54 and 86 - difference = 32
RBF at index 87 and 105 - difference = 18
BFR at index 88 and 106 - difference = 18

GCD : 2

From those repeated substring greatest common divisor could be drawn, which was 2. Although it didn't show the correct key size for this encrypted message those repeated substrings provide information to break the message. For example from the second and third substring, it can be said that the key was consist 2, 3, 6, 9 or 18. Let's see the second ciphered message produce from same message but with different key. Second ciphered message tested was this one:

L DD YJR BFUKO DZXZPVNTMGSON
HDEAUFWWJMEYNKCUIYWSLFAVMZMLGNLC
PDLCMGDDVBBSSPPKZPG HALEWWUKD VQRGIWITXFBBBPPAYGA
HFNJKOGPZSBKWBM

That message encrypted by 30 characters key. For this particular ciphered message there wasn't any repeated substring could be found. The main factor that helped these ciphered messages safe from Kasiski test was the size of the key. The key was long enough so it didn't repeated many times. Hence reduce the possibility of repeated substrings to appear. Beside that the fact that the whitespaces also encrypted in this proposed system, causing the ciphered message more durable against test that targeting polyalphabetic cipher.

6.4 Comparison Against Similar System (SIMPP)

For further understand the effectiveness of this proposed system, comparison with similar existing system must be performed. One of the existing systems that have been studied in this study was Secure Instant Messaging and Presence Protocol (SIMPP) developed using Jabber standard and implement Elliptic-Curve Cryptography (ECC) to secure the

communication [2]. Actually in order to compare the performance correctly, similar test must be performed as had been done to proposed system. However, since the source code or the application weren't available for public use, appropriate test couldn't be done.

SIMPP was proprietary of Taiwan Government because the project was partially funded by them. Although so, comparison still can be done by comparing the algorithm and protocol used by both systems.

In order to contrast its performance with proposed system properly, the comparison made were only in the process of encryption and not on the key exchange. This was done considering the scope of the project where the key exchange procedure was not considered. Comparison of this two systems shows on the Table 4 below.

Table 4: Comparison of Proposed System and SIMPP

	Proposed System	**SIMPP**
Cryptosystem	One Time Pad	AES (128 bits) with CBC mode
Key	Alphabet and whitespace (any size)	ECC *GF(p)*
Assumed Encryption and Decryption Process	Faster Encrypt one bit at a time	Slower Encrypt larger chunk of data (block)
Security	Secure Key always changed for every message	Secure Key size was big (128 bits) and AES is unbreakable until now
		Less Secure Same key is used for all transmission

Comparing two cryptosystem used by these systems was unfair. As shown in Table 4 proposed system was using OTP which is stream cipher and SIMPP was using AES which is a block cipher. Without further analysis it is known that stream cipher was faster compared to block cipher. It processes the data one bit at a time, while block cipher handle larger chunk of data hence require more memory and process time.

The key used in both systems also completely different. SIMPP got the key through ECC calculation on finding curve point over prime field. On the other hand, the proposed system's key was a combination of alphabet (A-Z) and whitespace. Complexity to obtain the curve point may increase the overall process time of the system.

Regarding the security of the ciphered message, both cryptosystem have great resistance against attack. AES reported until this day was unbreakable, however same key is re-used for entire conversation. There is a possibility that the message will provide a pattern that help guessing the key. OTP on the other hand, don't have such weakness as long as the golden rule of OTP of never reusing any key is obeyed.

7 Future Works

Study for implementing OTP in the IM system had been done. Nevertheless, there are several suggestions for any future works:

i. Testing with more than 2 clients simultaneously to see server responds

ii. Developing a better key generator which could provide key size secrecy.

iii. Extends the allowed characters for the message (symbols and numbers)

iv. Improving the key exchange mechanism

8 Conclusion

Study for the effectiveness of an IM system that implementing OTP encryption had been done. The proposed IM system was confirmed to be effective. Time taken to process each message that involved encryption process was quite fast, small increase compared when without encryption. It would not affect the whole system performance. Sizes of key used also not affecting the processing time, the message size being process did. The message confidentiality also proved secure, as long as the key was not re-used. Several suggestions for future works were stated.

References

[1] Gunter Ollmann, "Securing Against the "Threat" of Instant Messaging," Network Security Journal, vol. 2004, no. 3, pp. 8-11, March 2004.

[2] Chung-Huang Yang, Tzong-Yih Kuo, TaeNam Ahn, and Chia-Pei Lee, "Design and Implementation of a Secure Instant Messaging Service based on Elliptic-Curve Cryptosystem," Journal of Computers, vol. 18, no. 4, pp. 31-38, January 2008.

[3] M. Day, J. Rosenberg, and H. Sugano, "A Model for Presence and Instant Messaging," RFC - 2778, pp. 1-17, February 2000.

[4] N. Leavitt, "Instant Messaging: A New Target for Hackers," IEEE Computer Society, pp. 20-23, July 2005.

[5] Behrouz A. Forouzan, Cryptography & Network Security. New York, US: McGraw-Hill, 2008.
[6] Steven M. Bellovin, "Frank Miller: Inventor of the One-Time Pad," Cryptologia, vol. 35, no. 3, pp. 203-222, July 2011.
[7] Shannon C. E., "Communciation Theory of Secrecy System," Bell System Technical Journal, vol. 28, no. 4, pp. 656-715, October 1949.
[8] Pidgin-Encryption. (2011, October) Pidgin-Encryption. [Online]. http://pidgin-encrypt.sourceforge.net/