Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 10, October 2014, pg.164 – 171

RESEARCH ARTICLE

Secure SMS Based on Internet Service

Saja T. Ahmed[1], Loay E. George[2]

[1]College of Science/ University of Baghdad/ Iraq

[2]College of Science/ University of Baghdad/ Iraq

[1] sajataha@ymail.com; [2] loayedwar57@scbaghdad.edu.iq

Abstract— In the past few years a new generation of mobile messaging systems have been developed in conjunction with the explosive growth in the use of smart phones. As this technology enabling for accessing Internet based services, the new-generation the messaging applications was directed to play the role of traditional text messaging(SMS). Such applications require only the user's phone number for registration. Unfortunately, these appl1icationsdo not provide a built-in support for any security feature. Sometime, the basic security features (e.g., authentication and confidentiality between the communicators) must be ensured in messaging service. In addition to that, most of the modern communications via internet takes place "through the air"; therefore the risk of interception becomes greater than in case of wired networks. If the message is not encrypted, or encrypted using a weak algorithm, the attacker can get it and may make hard harm. This paper presents a security solution to ensure confidentiality and authentication of exchanged messages, the users of proposed system can communicate exclusively via web server and by using a dedicated android application. The exchanged messages are encrypted using stream cipher RC4 with a personalP2P dynamic encryption key.

Keywords— SMS Security, Hash Function, Dynamic Key, Avalanche effect

I. INTRODUCTION

Huge penetration of mobile devices, in particular smart phones, and the development of mobile broadband are important factors in the development of Mobile Social Networking applications and messaging services. At the end of 2008 close to half of the world's population had a mobile phone and about 54% of telecom revenue came from mobile services [1]. According to portio research (in its mobile fact book 2013) worldwide smart phone shipments will reach 1,095 billion by 2016 and accounts for %51 of the total handset shipments, this is a huge growth (taking into consideration it was 175million at 2009; which represents%15 of the total handset shipments at that time). This indicating massive growth in the bandwidth capacity of the mobile network. In addition to this, the smart phones will also have access to other high bandwidth networks like WiFi.

Among smart phones devices such as the iPhone and IOS, Google Android has been gaining popularity and it has about 75% market share in the mobile operating system market, with a 91.5% growth rate over the past year [2]. The story of Android relies on the development of the free and open source OS built around the Linux kernel. The Linux kernel itself started life in 1991, and it was built on earlier free operating system initiatives (e.g. GNU, BSD and Minix). The open nature and low cost of Linux have made it popular as an embedded operating system for electronic devices including smart phones [3]. With Android phones being ubiquitous, they become a worthwhile target for security and privacy violations, in addition to that; using Android based smart phones applications and internet; people are sharing information with other people but they are not sure that information is securely transmitted or not [4]. In general, Internet messaging application offer free calls and text messages to other subscribers, providing an Internet-based alternative to the traditional communication methods managed by cellular network carriers such as SMS, MMS and voice calls, and although internet messaging application's users are estimated to be multi millions, but very little attention has so far been paid to the security measures (or lack thereof) implemented by these provider [5].

Schrittwieser et al [5] have analyzed nine popular mobile messaging and VoIP applications (such as whatsApp and Viber) evaluate their security models with a focus on authentication mechanisms, they found that a majority of the examined applications use the user's phone number as a unique token to identify the accounts, no additional authentication mechanisms other than the phone number are used by these applications, which further encumbers the security barriers. Thus, such applications are vulnerable against hijacking attacks. Also, most of the applications suffer from other vulnerabilities (such as account enumeration). Also, they indicated that all the identified flaws are stem from some well-known software errors which occur during the design and implementation phase. Although the assigned vulnerabilities may not endanger human lives, but they may have severe impacts on the privacy of millions of users.

This paper mainly introduce the idea of using dynamic encryption key that depending on time stamp beside to using stream cipher RC4 algorithm to encrypt messages between communicators. The proposed system uses a web server for accepting and forward encrypted messages to the main server. The clients' application is developed using Java programming language and Eclipse SDK and, working on Android environment. The applications on HTTP and GPRS servers have been developed using Apache web server and PHP scripting.

II. Proposed Method

This paper proposes an improvement to the encryption scheme that presented by [6]; the previous scheme uses convolution and shuffling methods to meet diffusion and confusion, then message is encrypted using RC4 with static encryption key. In this paper, the dynamic encryption key is generated using a proposed hash function. The encryption and decryption processes along with generating dynamic key are explained in Figure (1). The scheme stages are as follows:

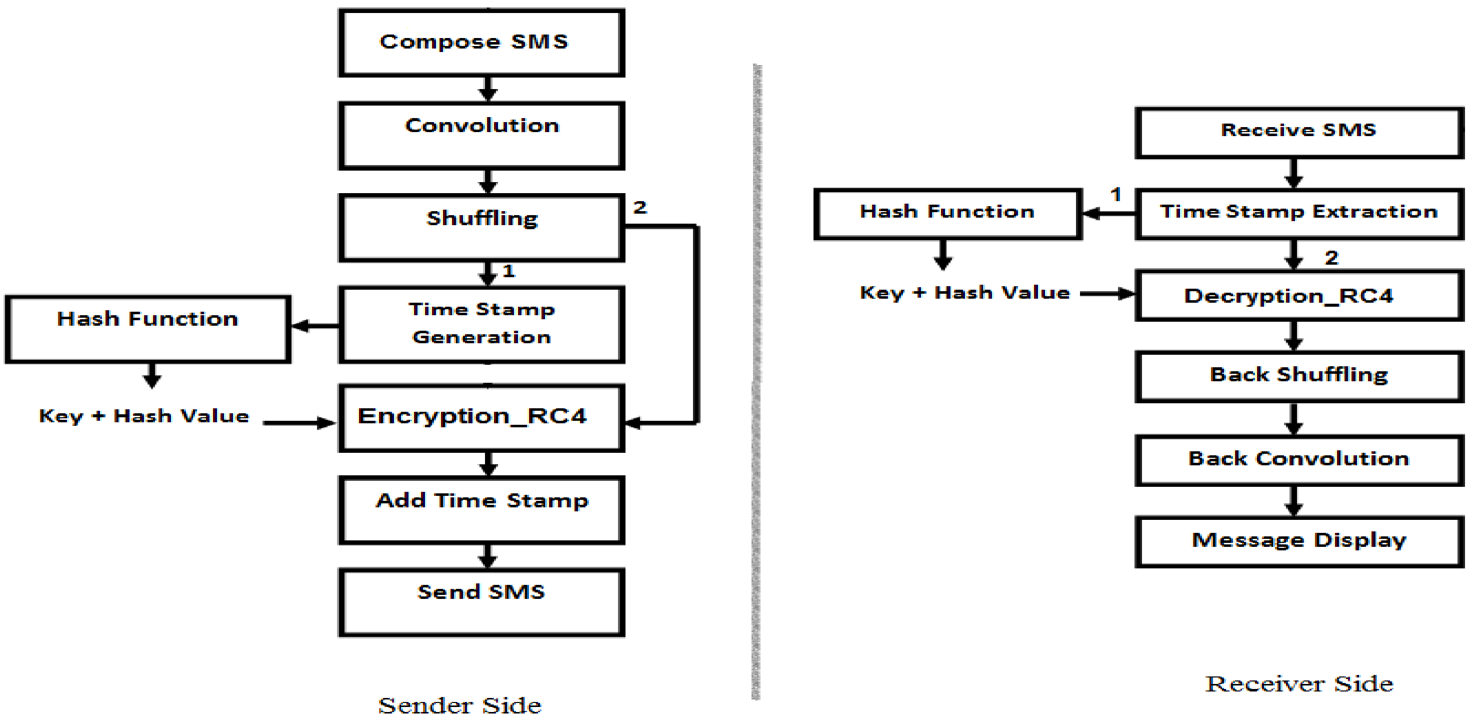

Fig.1 the proposed methods at sender and destination client

1. *Encryption*: the convoluted and shuffled message is encrypted using RC4 method. Each user has a dedicated P2P RC4 key to exchange encrypted messages with the HTTP server. This encryption step provides both message confidentiality and authentication. Each RC4 encryption key consists of 128-bit with adding proposed hash value that is produced depending on a time stamp. The time stamp is generated from the time and date at the moment of encrypting and sending it. The time stamp constituted as follows: the minutes value are shifted to left by six binary digits (because the maximum minute value is equal to 59 which takes 6 bit to represent it), and adding it with the value of seconds, then the result is added to the hour value after shifting it to left by twelve binary digits. Then, the result is added to day number value after shifting it left by seventeen binary digits. Then whole result is added to the least decimal digit of year (by making mod operation on the year value by 10 and shift the results to left by twenty six binary digits:

yr = Get Current Year % 10
dy = Get Current Day
hr = Get Current Hour
mn = Get Current Minute
sc = Get Current Second
TimeStamp = sc + (mn<< 6) + (hr<< 12) + (dy<< 17) + (yr<< 26)

This equation gives the time stamp which is consist of 32 bits. Time stamp bytes are convoluted, and the bits of convoluted bytes are shuffled using the proposed algorithms in [6]. Then, the time stamp is entered into the proposed hash function to generate a hash value that concatenated with the encryption key to generate a flying key. This makes the same message is, every time, encoded differently because used dynamic encryption key would be different. The introduced hash function is designed to be simple and can generate a hash value its length is variable (it can be 8, 10 bits or higher), but in the conducted tests of this work it is taken 10 bits. Step1 takes the time stamp and converts it to binary array. Step2 produce one bit from every 3 bits, by applying XOR convolution, till getting the 10 bits hash value using XOR operator.

```
The Proposed Hash Function
Goal:  Make encryption key  different depending on time stamp
Input: Time//Time Stamp as integer
Output: Hash Value as integer

Step1: ///Convert Time to array of binary.
set Hash ← 0
set d ← 0
set Tb ← Convert time to array of bytes
set Tbits ← Convert Tb to binary array

Step2: Pick up one bit from every three bits
for all i Do {where 0 ≤ i<Tbits.Length - 2} increment i by 3
    set h ← Tbits[i] ^ Tbits[i +1] ^ Tbits[i +2]
    Hash = Hash + (h*(2<<d));
    Increment d by one
end for
Return Hash
```

RC4 algorithm was applied on android client mobile device as a class contains methods which can be called as follows:

a. *RC4 constructor*: this defines a new instance of RC4 class. It should be called before any other method to ensure the passing of encryption key to RC4.
b. *RC4.encrypt*: it is called to perform message encryption. This method takes as input the plain text as an array of characters and returns the cipher text as an array of characters too.

After completing the encryption at sender side, the generated time stamp is attached to the encrypted message as a plain text (i.e., without encryption) to allow receiver to extract the time stamp from the received message and entered into the same hash function to generate same hash value that concatenated with decryption key to retrieve the original message. The secure message payload became as illustrated in Figure (2).

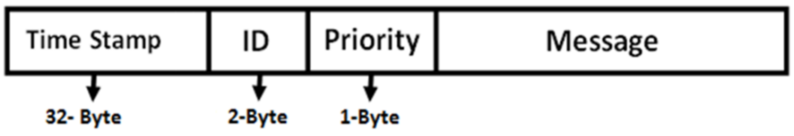

Fig.2 secure message payload

2. Decryption: the decryption process begins with the step of timestamp extraction (which is done after obtaining the encrypted message), the time stamp part should extracted and enter it into the proposed hash function (that used at sender side) to generate the hash value. The hash value is concatenated with the key to get the same flying encryption key for the sender. The RC4 algorithm is applied on recipient mobile device as a class contains methods some of which are clarified earlier, when describing the encryption module at sender side. Now, at the destination client the incoming SMS should decrypted using RC4.decrypt method after getting the encryption key. Then the back shuffling and back convolution algorithms are called to get plain message as described in [6].

III. SYSTEM WORK FLOW

The sender client can send his message using a dedicated Android application. If the sender android device stops working for any reason (for example: loss of charging or disoperation), then he/she can access the web server by using his Email and a password for authentication, through this web utility the sender can send the required message as secure as he/she is using android device. The web server is established using Apache web server. PHP is used as server-side HTML embedded scripting. The process of sending secure message via web server is clarified in figure (3); it implies the following steps:

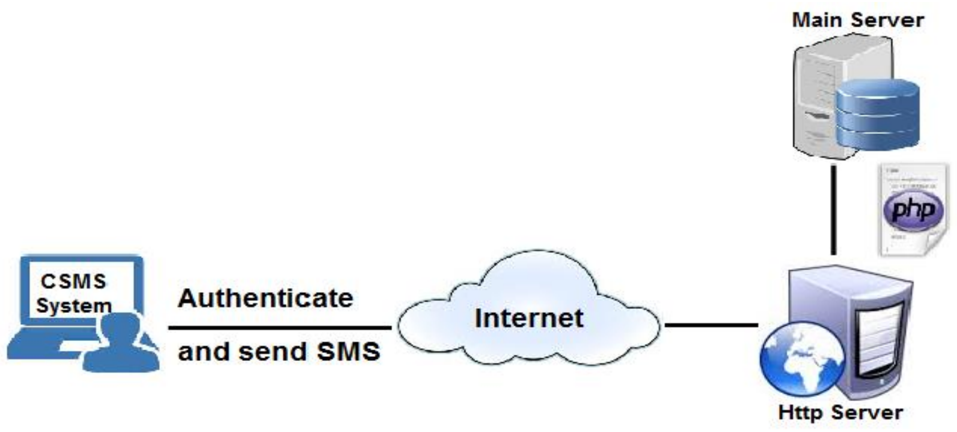

Fig. 3 SMS Sending using Web Server

1. The sender must authenticate himself by using an email; he/she should feed his password through a dedicated HTML page. The embedded PHP script will compare the input data with predefined subscribers data stored in mysql database that installed on the web server.
2. After passing the authentication stage successfully then another HTML web page (holds embedded PHP) will be displayed to allow the sender client enters his message. SMS priority and the destination ID should assigned by sender, the entered values are attached together and encrypted in same way as in the android device and sent to main server.
3. In order to forward message from web server to main server, a special PHP script is developed to open a dedicated socket between the web server and main server to process the incoming messages from web.

I. Test Results

The avalanche property is a desirable property in cryptographic hash function; it refers to the effect of slight change in the input (e.g., one bit) the whole cipher text output; such that it must be changed significantly (e.g., nearly half of the cipher bits). If encryption process does not exhibit the avalanche effect to a significant degree, then it is considered unacceptable (or poor) randomization process; because in such case the cryptanalyst can make predictions about the input, once he/she is able to get samples of the plain and cipher texts output. If the encryption system use only one bit hash value, either 0 or 1, the cipher result will be two different cipher texts for same message; and if two bits hash value is added to the encryption key then the system can produce four different cipher texts and so on. A 16-bit hash value is added with encryption key to have the possibility of producing 65536 variant for the plain text when using same encryption key. The produced cipher text will be different at every encryption process instant because the generated dynamic keys due to adding 16 bits hash value depending on the timestamp. Table (1) presents the NIST randomness tests results for the message "***hello server im in Baghdad***" encrypted at different times of day using the encryption key "***ab78495fdjk6210d***" along with auto generated hash value for eight different times.

TABLE1

AVALANCHE EFFECT RANDOMNESS TEST

No	***Cipher Text***	***Hash value***	***Randomness test***
1	1d6dcb929e516b2738cfae1b0a07735ed7730462bb5bba8de77d	1	Frequency test
			0.1655 : pass
			Block test
			0.5760 : pass
			Run test
			0.3299 : pass
2	c7e8ca0b325139fb198cd5f5cde17ef2cdc8bf9db50a2e5b9318	3	Frequency test
			0.2673 : pass
			Block test
			0.5760 : pass
			Run test
			0.9316 : pass
3	f0c14197f029c10c1a1621d1326b1556fbd1440893146fcdbeb2	0	Frequency test
			0.1655 : pass
			Block test
			0.3585 : pass
			Run test
			0.8844 : pass
4	27a1e409c46f4794b5a58bd145d3e5f7c27984928f15a8917333	6	Frequency test
			0.7815 : pass
			Block test
			0.8758 : pass
			Run test
			0.5753 : pass
5	7ef44a701bf4112a3ef61468e7002f6ec1525f47769e3aa31c7c	4	Frequency test
			0.8897 : pass
			Block test
			0.3835 : pass
			Run test
			0.3323 : pass
6	e5c255cc4899b0ba8593adf6c932e85d5aa91244ec90272618c7	2	Frequency test
			0.3317 : pass
			Block test
			0.8932 : pass
			Run test
			0.0606 : pass

7	b2fb85792d70b63d821b3d1a6ab791d8c5046bf43302b165482b	5	Frequency test
			0.6774
			Block test
			0.5474
			Run test
			0.3253
8	ec0cef81afe83855b091b567ee52191e4bf7d348796bd3d2c192	7	Frequency test
			0.5791
			Block test
			0.4631
			Run test
			0.7976

Table (2) shows the avalanche effect impact, the results illustrate that around %50 of the bits in the cipher text (where, the cipher text length is 208 bits) are different, the avalanche effect is pronounced after just one bit of the key is changed (where the key length is 128 bit with 16 bit hash value).When 2 bits of the key are changed, the two cipher texts differ in 121 bit positions, and when 5 bits of two keys are changed, the two cipher texts differ in 105 bits.

TABLE2

AVALANCHE EFFECT TEST

#	**Key**	**Δ**	**Cipher key**	**§**
1	ab78495fdjk6210d00	1	f0c14197f029c10c1a1621d1326b1556fbd1440893146fcdbeb2	94
	ab78495fdjk6210d01		1d6dcb929e516b2738cfae1b0a07735ed7730462bb5bba8de77d	
2	ab78495fdjk6210d01	2	1d6dcb929e516b2738cfae1b0a07735ed7730462bb5bba8de77d	121
	ab78495fdjk6210d02		e5c255cc4899b0ba8593adf6c932e85d5aa91244ec90272618c7	
3	ab78495fdjk6210d03	3	c7e8ca0b325139fb198cd5f5cde17ef2cdc8bf9db50a2e5b9318	95
	ab78495fdjk6210d04		7ef44a701bf4112a3ef61468e7002f6ec1525f47769e3aa31c7c	
4	ab78495fdjk6210d07	4	ec0cef81afe83855b091b567ee52191e4bf7d348796bd3d2c192	96
	ab78495fdjk6210d08		db2a1d9fc3406c3eecc053c5fe73bc0d5c7659eab850ef6c9d05	
5	ab78495fdjk6210d65	5	36253d909955a19e180e4c011c34d9088eb74410843db7c64d51	105
	ab78495fdjk6210d16		76fe5235a6702c1e0e027763069824782a74486a4aca431e148d	

The computational load consumption of the implemented encryption operations must be taken into account when developing any encryption scheme. In our proposed system asymmetric encryption is avoided because of its high computational overhead; so, the proposed SMS encryption system is a symmetric cryptographic method. It is good for mobile devices due to their limited resources (i.e., less computing power, insufficient memory and limited power energy). The secure messaging application that used in proposed system for clients was developed using Java programming language to work in Android platform. It had been tested on GT-I9300 (GALAXY SIII) device with 4GHz Quad Core Processor and 16GB memory. The test aimed to evaluate the resources usage performance of the developed application. Figure (4) illustrates the CPU usage of the application during a time interval extended to 180 seconds.

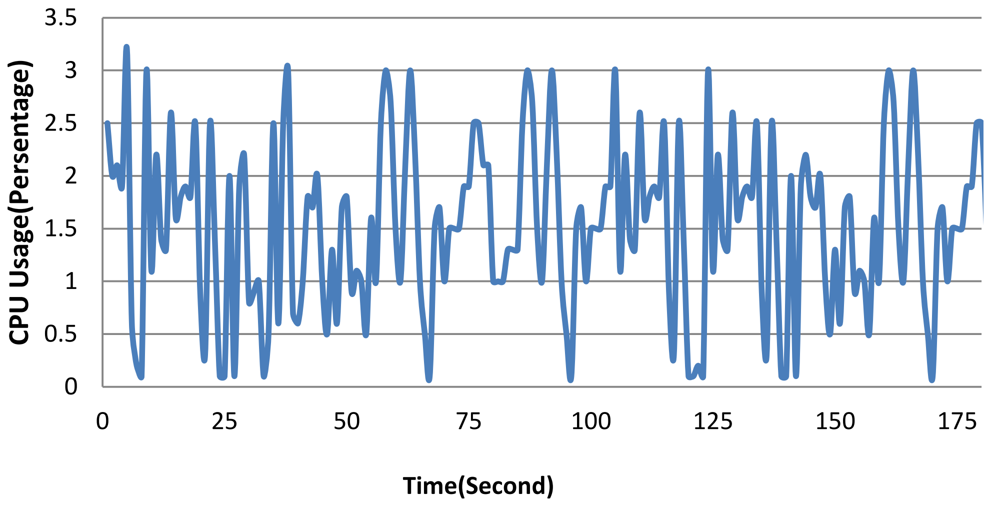

Fig. 4 Percentage of CPU Usage Time of Client

Also, the memory usage had been monitored. Figure (5) shows the memory usage of the client application.

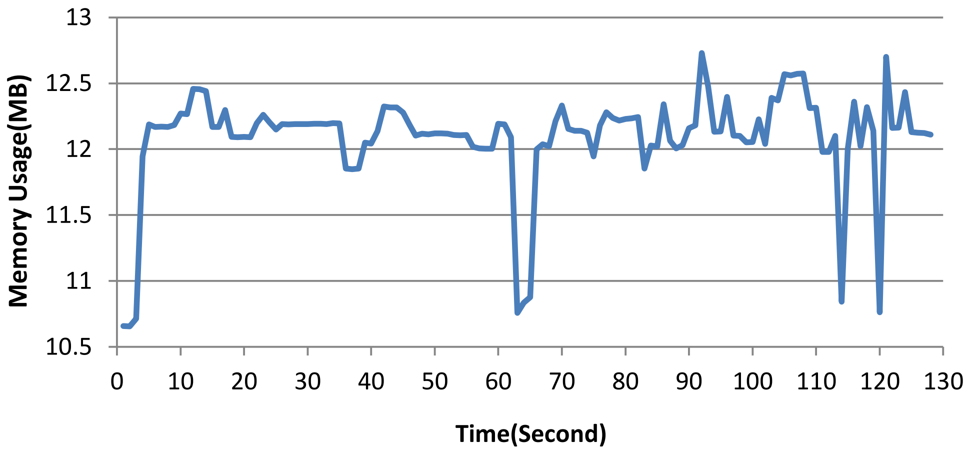

Fig. 5 Memory Usage of Client

The application uses 0.1 to 3.2% of the processor time, and as average it uses 1.51% of the processor time. Figure (6) shows the client application (com.SMS) percentage of processor usage time along with other running applications.

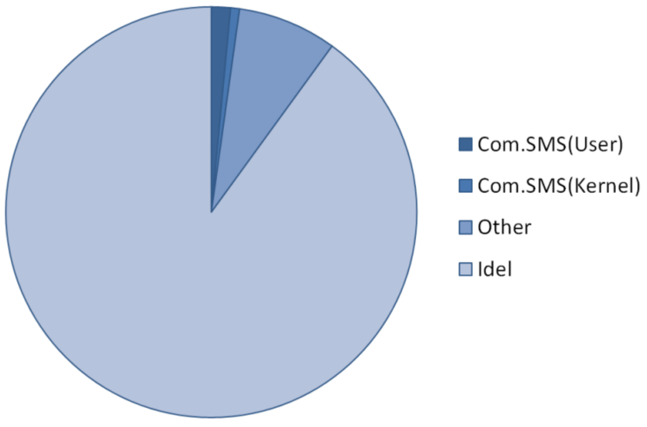

Fig.6 Total Processor Usage Time

REFERENCES

[1] Hammershoj, A.; Sapuppo, A.; Tadayoni, R.; "*Mobile Platforms-An analysis of Mobile Operating Systems and Software development platforms*", CMI international conference on social networking and communities 25-26 November 2009. Copenhagen, Denmark, 2009.

[2] Fritz, Ch.;Arzt, S.;Rasthofer, S.;Bodden, E.;Bartel, A.; Klein, J.;Traon, Y.;Octeau, D.; McDaniel. P.; "*Highly Precise Taint Analysis for Android Application*", Technical Report TUD-CS-2013-0113., available on: bodden.de/pubs/ TUD-CS-2013-0113.pdf, 2013.

[3] Tilson,D.; Sorensen, C.; Lyytinen, K.; "*Change and Control Paradoxes in Mobile Infrastructure Innovation: The Android and iOS Mobile Operating Systems Cases*", System Science(HICSS), 45th Hawaii International Conference on SystemSciences, 2012.

[4] Sushant, A.P.;Pravin, D.S.; "A *Survey on Instant Message and Location Sharing System for Android*", International Journal of Application or Innovation in Engineering & Management (IJAIEM), ISSN: 2319 – 4847, Volume 2, Issue 10, Pp.219-221, 2013.

[5] Schrittwieser, S.; Fruhwirt, P.; Kieseberg, P.; Leithner, M.; Mulazzani, M.; Huber, M.; Weippl, E.; "*Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications*", in Network and Distributed System Security Symposium (NDSS 2012), 2012.

[6] Saja, T. Ahmed, Loay E. George, "*Secure Messaging System over GSM Based on Third Party Support*", International Journal of Engineering and Innovative Technology (IJEIT), ISSN: 2277-3754, Volume 4, Issue 2, Pp.27-32, 2014.