ELSEVIER

Computer Communications 24 (2001) 1090–1096

computer communications

www.elsevier.com/locate/comcom

Short Communication

Stream ciphers for GSM networks

Chi-Chun Lo[a,*], Yu-Jen Chen[b,1]

[a]Institute of Information Management, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC
[b]Department of Information Management, Chang-Gung University, 259 Wen-Hwa 1st Road, Kwei-Shan, Tao-Yuan 333, Taiwan, ROC

Received 14 March 2000; accepted 29 August 2000

Abstract

With the advance of wireless communications technology, mobile communications has become more convenient than ever. However, because of the openness of wireless communications, how to protect the privacy between communicating parties is becoming a very important issue. In this paper, we focus on the security of transferring voice message on the Global System for Mobile communication (GSM) networks. Stream cipher is recommended for message encryption and decryption. First, a key generator is presented. Then, on the basis of the key generator, stream ciphers are designed with respect to different levels of securities of GSM networks. Simulation results indicate that the key generator always produces key strings of evenly distributed 0's and 1's and with a very long period; i.e. it is very unlikely that repeated patterns will appear. Cryptanalysis and operational analysis show that the proposed stream ciphers are secure and efficient; consequently, they provide a comprehensive set of message encryption/decryption algorithms for GSM networks.

Keywords: GSM networks; Wireless communications; Mobile communications; Stream cipher; Linear feedback shift register

1. Introduction

Mobile communications has become more popular and easier for the past few years. Nowadays, people can communicate with each other on any place at any time. However, the openness of wireless communications poses serious security threats to communicating parties [8]. How to provide secure communication channels is essential to the success of a mobile communication network [7]. The Global System for Mobile communications (GSM) is the standard for digital mobile communications. It has a comprehensive set of security features [5]. The GSM standards suggest a proprietary algorithm for message encryption and decryption while in this paper, we recommend that a non-proprietary stream cipher be used instead. A key generator (KG) is first introduced; then, stream ciphers are proposed by using the KG as their basic building block. Simulation results indicate that the KG can always produce key strings of evenly distributed 0's and 1's and with a very long period; i.e. it is very unlikely that repeated patterns will appear. Cryptanalysis and operational analysis show that the proposed stream ciphers are secure and efficient. In the following section, literature review is presented. Section 3 describes the KG followed by the design of the proposed stream ciphers. Security and efficiency analyses are given for each proposed stream cipher. Section 4 discusses how different stream ciphers can be used to satisfy the security requirements of GSM networks. Section 5 concludes this paper with possible future research directions.

2. Literature review

2.1. GSM [5,9–12]

The GSM is the first mobile communication system which has comprehensive security features [6]. It is gaining tremendous supports from the telecommunication industry. The security architecture of GSM is intended to prevent unauthorized network access, disallow subscriber impersonation, protect confidentiality, and provide privacy. GSM's security services include anonymity, authentication, signaling protection, and user data protection.

Fig. 1 depicts the security architecture of GSM. It is composed of three tiers. The first tier — namely, the A3 algorithm, uses the challenge–response method [1] for user authentication. The second tier — namely, the A8 algorithm, uses the output from the A3 algorithm to generate the key string for the A5 algorithm. The last tier — namely,

* Corresponding author. Tel.: +886-3-5731909; fax: +886-3-5723792.
E-mail addresses: cclo@cc.nctu.edu.tw (C.-C. Lo), cyr@mail.cgu.edu.tw (Y.-J. Chen).
[1] Tel.: +886-3-3283016, ext. 5823; fax: +886-3-3271304.

0140-3664/01/$ - see front matter
PII: S0140-3664(00)00333-9

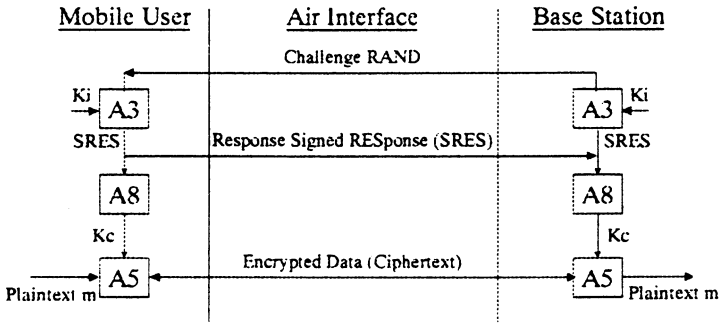

Fig. 1. Security architecture of GSM.

the A5 algorithm, uses a proprietary algorithm for message encryption/decryption. This architecture has the following problems: challenge–response is a simple authentication method, but is not very secure [4]; the length of the key generated by the A8 algorithm is fixed (114 bits), hence this key might be disclosed by the brute-force attack [1]; the A3, A8 and A5 algorithms are proprietary, thus their security cannot be easily verified [3]; and the A5/1 algorithm is subject to export control.

2.2. Stream cipher [1,13,14]

Symmetric cryptosystems are further classified into block cipher and stream cipher. Block cipher divides the plaintext into blocks and encrypts each block independently. As for stream cipher, it encrypts the plaintext on a bit-by-bit (or byte-by-byte) basis. In general, stream cipher is much faster than block cipher [1,14]. GSM networks carry mostly voice, which is one type of continuous data. Since stream cipher is based on the eXclusive OR (XOR) operation, to encrypt/decrypt voice data bit-by-bit (or byte-by-byte) using stream cipher is very efficient. Therefore, stream cipher is a good choice for message encryption/decryption for GSM networks.

The major problem of stream cipher cryptography is the difficulty of generating a long unpredictable bit pattern (keystream). How to increase the length of the keystream while still maintaining its randomness is important to the security of stream cipher. In the one-time pad of stream cipher, a keystream is a sequence of randomly generated bits, and the probability of one bit to be 1, independent of other bits, is equal to one half. An ideal keystream in one-time pad is purely random and has infinite length. The keystream cannot be generated by the receiving end, and cannot be distributed to the receiving end either. The pseudorandom bit generator [14] has been widely used to construct the keystream. It generates a fixed-length pseudorandom noise as the keystream. A pseudorandom bit generator consists of one or more linear feedback shift registers (LFSRs) [1].

3. Stream ciphers

Since the A5 algorithm is proprietary and the A5/1 algorithm is subject to export control, we propose a new GSM security architecture, as shown in Fig. 2, in which the C5 algorithm replaces the A5 algorithm. The C5 algorithm uses stream cipher for message encryption and decryption. It takes a keystream M from A8 as its initial keystream. It processes input message m on a byte-by-byte basis. First, a KG is introduced. Then, on the basis of the KG, three stream ciphers — namely, $S1$, $S2$, and $S3$, are designed with respect to different levels of securities of GSM networks.

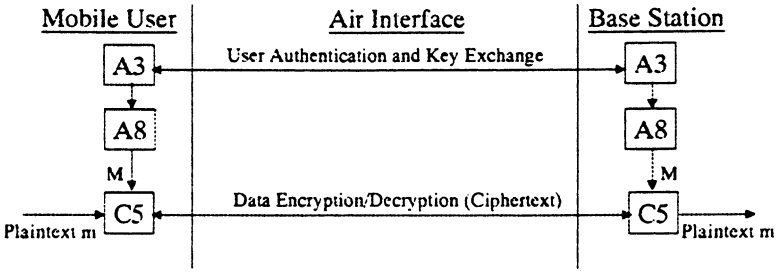

Fig. 2. The proposed security architecture of GSM.

Table 1
Simulation results of $M1$ and $M2$ (a/b, where a is the probability of being 0 and b is the probability of being 1)

	m			
	Type-1	Type-2	Type-3	Type-4
(a) M1				
Type-1	0.500031/0.499969	0.500014/0.499986	0.500014/0.499986	0.500014/0.499986
Type-2	0.499762/0.500238	0.503766/0.496234	0.503766/0.496234	0.503766/0.496234
Type-3	0.500586/0.499414	0.484420/0.515580	0.484420/0.515580	0.484420/0.515580
Type-4	0.501155/0.498845	0.510001/0.489999	0.510001/0.489999	0.510001/0.489999
(b) M2				
Type-1	0.500015/0.499985	0.499992/0.500008	0.500006/0.499994	0.500001/0.499999
Type-2	0.498557/0.501443	0.499939/0.500061	0.500351/0.499649	0.497654/0.502346
Type-3	0.499162/0.500838	0.486187/0.513813	0.483608/0.516392	0.483438/0.516562
Type-4	0.499078/0.500922	0.507109/0.492891	0.506546/0.493454	0.499536/0.500464

3.1. Key generator (KG)

3.1.1. Design

For an initial keystream of length k bytes, $(M_0, M_1, \ldots, M_j, \ldots, M_{k-1})$, and an input message of length n bytes, $(m_0, m_1, \ldots, m_i, \ldots, m_{n-1})$, the keystream of length n bytes, $(Kc_0, Kc_1, \ldots, Kc_i, \ldots, Kc_{n-1})$, is generated according to the following procedure:

Step 1. Let $i = 0$, $j = 0$, $N = 170$, $cc = 0$;
Step 2. $M_j = M_j + cc$;
If $j = 0$ then $M_j = M_j$ XOR N; else
$M_j = M_j$ XOR M_{j-1};
Step 3. $Kc_i = M_j$; $N = (N + m_i) \bmod 256$;
$i = i + 1$; $j = j + 1$; $cc = cc + 1$;
If $j = k$ then reset j to 0;
If $cc = 256$ then reset cc to 0;
If $i = n$, then exit; else goto Step 2.

where N, an eight-bit string, is used by the XOR operation to change M_0 at the beginning of each cycle of M; cc is a number added to M_j ($j = 0$ to $k - 1$) so as to increase the randomness of M_j.

Note that the initial value of cc and N can be any number between 0 and 255 ($2^8 - 1$). In the KG, we use the number 170 (10101010_2) as the default value of N and 0 as the default value of cc.

Table 2
Keystream periods with respect to pairs 1 to 4

M	m			
	Type-1	Type-2	Type-3	Type-4
(a) Pairs 1 and 2				
Type-1	102400	102400	102400	102400
Type-2	102400	102400	102400	102400
Type-3	102400	102400	102400	102400
Type-4	102400	102400	102400	102400
(b) Pairs 3 and 4				
Type-1	130000	130000	130000	130000
Type-2	130000	130000	130000	130000
Type-3	130000	130000	130000	130000
Type-4	130000	130000	130000	130000

3.1.2. Security analysis

The security of the KG is evaluated by simulation. The simulator is coded in C language and is running on an Intel Pentium II-266 MHz PC with 64 MB RAM. Two metrics, randomness and period, are selected for security evaluation. *Randomness* is defined as the probability of being 0 or 1. For a purely random keystream, the probability of one bit to be 1, independent of other bits, is equal to one half. *Period* is defined as the number of bytes in the repeated pattern of a keystream; for example, if a keystream of n bytes has no repeated pattern, then its period is equal to n.

In the following simulations, we consider four different types of initial keystreams (Ms). Type-1 M is purely random, Type-2 M is of all 0's, Type-3 M is of all 1's, and Type-4 M consists of alternating 8-bit 0's and 8-bit 1's. Four different types of input messages (ms) are also considered. Type-1 m is purely random, Type-2 m is of all 0's, Type-3 m is of all 1's, and Type-4 m consists of alternating 8-bit 0's and 8-bit 1's. There are total 16 (4×4) combinations (cases) with respect to Ms and ms.

3.1.2.1. Randomness. Two different sizes of Ms, 23040 bytes ($M1$) and 23000 bytes ($M2$), are assumed. Note that 23040 is a multiple of 256 (2^8) whereas 23000 is not. The size of the input message is 2000 KB. Each test case is run 100 times, and then the average is computed.

Table 1 shows the simulation results of $M1$ and $M2$, respectively.

By examining Table 1(a), we notice that the probability of being 0 and that of being 1 are the same, for Type-2 m, Type-3 m, and Type-4 m. This phenomenon indicates that the length of the initial keystream M should not be multiples of 256, since the KG processes data on a byte-by-byte basis. By observing Table 1(b), we find that the probability of being 0 and that of being 1 are close to one half, for all test cases. This phenomenon indicates that the KG does

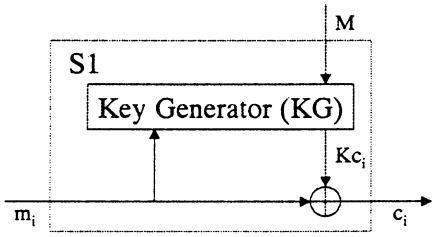

Fig. 3. The stream cipher *S*1.

produce key strings of evenly distributed 0's and 1's, regardless of input patterns.

3.1.2.2. Period. We consider four different pairs of (l_M, l_m), where l_M and l_m represent the length (the number of bytes) of the initial keystream M and the input message m, respectively. Those four pairs are (2400, 102400), (4800, 102400), (2400, 130000), and (4800, 130000). For Type-1 M and m, their period is equal to l_M and l_m, respectively. For Type-2 M and m, their period is equal to 1. For Type-3 M and m, their period is equal to 1. For Type-4 M and m, their period is equal to 2. Sixteen combinations are tested for each pair. Table 2 presents the simulation results.

By examining Table 2, we notice that keystreams generated by the KG always maintain the maximal period, regardless of input patterns. This phenomenon indicates that the KG is able to produce key strings with a very long period.

3.1.2.3. Discussion. In the KG, we use the input message, (m_i), as a random seed to increase the period of the initial keystream M. However, one may have the following concern: "Will this random seed affect the security of the output keystream?". By using simulation, we have tested all possible values of the random seed, 0–255. Simulation results indicate that there are no repeated patterns. Randomness holds for every possible value of the random seed.

3.1.3. Efficiency analysis

The KG is very efficient since it only uses three primitive operations: XOR, ADD, and INC.

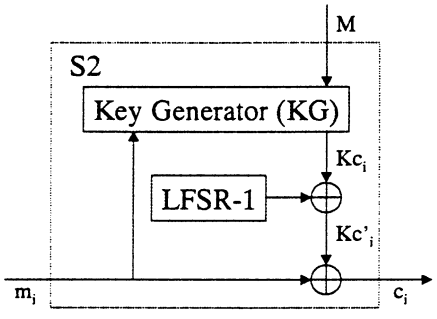

Fig. 4. The stream cipher *S*2.

3.2. Stream cipher 1 (S1)

3.2.1. Assumption

The initial keystream M is random, and control parameters, M, N, and cc, are securely protected.

3.2.2. Design

The stream cipher *S*1, as shown in Fig. 3, consists of the KG and an XOR operator. In *S*1, the ciphertext, (c_i), is obtained by using the XOR operation between the input message, (m_i), and the output keystream, (Kc_i), of KG.

3.2.3. Security analysis

*S*1 is as secure as the KG, since the additional XOR operation required by *S*1 does not introduce any security threat.

3.2.4. Efficiency analysis

*S*1 is as efficient as the KG, since the additional XOR operation required by *S*1 only consumes a small amount of CPU time.

3.3. Stream cipher 2 (S2)

By closely examining the output keystream, (Kc_i), of the KG, we notice if the initial keystream M is not purely random, then the output keystream appears to have some regularities, despite that it is a binary string of evenly distributed 0's and 1's and with infinite period. For example, for a purely random input message m (Type-1 m) and a non-random initial keystream M (Type-4 M), M is given as follows:

00 FF 00 FF 00 FF 00 FF 00 FF 00 FF

00 FF 00 FF 00 FF 00 FF 00 FF 00 FF, (1)

then the first 72 bytes of the output keystream can be shown as follows:

AA AA A8 AA AE AA AC AA A2 AA A0 AA

A6 AA A4 AA BA AA B8 AA BE AA BC AA

1F DC 1E DB 11 D6 1C D5 17 DC 1E D3

19 D6 1C CD 2F FC 1E CB 21 F6 1C C5

5E 53 03 0D 48 43 11 1D 52 47 1F 11

44 57 0D 01 6E 53 33 3D 58 63 01 0D. (2)

By examining Eq. (2), we have the following observations: byte "AA" appears 13 times in Eq. (2), and 14 bytes in Eq. (2) have the hexadecimal number "1" as their leftmost four bits.

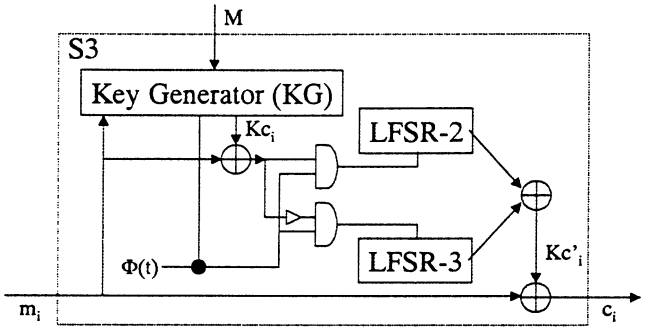

Fig. 5. The stream cipher *S*3.

3.3.1. Assumption

The initial keystream *M* is not random, and control parameters, *M*, *N* and *cc*, are securely protected.

3.3.2. Design

To remedy the "non-random *M*" problem, we propose the stream cipher *S*2. *S*2, as shown in Fig. 4, consists of the KG, the LFSR-1, and two XOR operators, where the LFSR-1 is a maximal-length LFSR, i.e. a primitive polynomial mod 2, of length 32 bytes. The tap sequence of the LFSR-1 is (32,7,6,2,0). The initial state (a byte string) of the LFSR-1 is randomly generated with the exception of the rightmost byte. The rightmost byte is always set to 255 ($0 \times FF$) to avoid the possibility that the initial byte string is of all 0's.

The stream cipher *S*2 is stated as follows:

Step 1. The keystream, (Kc_i'), is obtained by using the XOR operation between the output keystream, (Kc_i), of the KG, and the output keystream of the LFSR-1.
Step 2. The ciphertext, (c_i), is obtained by using the XOR operation between the input message, (m_i), and the keystream, (Kc_i').

3.3.3. Security analysis

The security of *S*2 is enhanced by the maximal-length LFSR. By applying *S*2 on Eq. (1), we obtain the keystream, (Kc_i'), of which the first 72 bytes are as follows:

59 C4 0D 20 A2 A0 8D EA AF DE 81 24

0E 23 10 3A 8D B4 42 DE E0 EF 5D CC

80 7E 7D 1C A0 C8 10 2C AB A4 A1 19

8C CD 61 9C 37 E1 2A 7E F9 14 44 B1

7E D2 B7 0E 2B B1 37 9B F4 C7 A5 46

42 A0 E4 5A AB 85 DE C8 27 68 8E D0. (3)

Apparently, there is no regularity existing in Eq. (3). *S*2 indeed fixes the "non-random *M*" problem.

3.3.4. Efficiency analysis

The LFSR-1 requires three XOR operations and one SHIFT operation. *S*2 consumes more CPU time than *S*1, which requires only an additional XOR operation. However, both XOR and SHIFT are primitive operations; hence, *S*2 is still considered to be efficient.

3.4. Stream cipher 3 (S3)

In *S*2, if control parameters, *M*, *N* and *cc*, and the initial state of the LFSR-1 are known by an attacker, the attacker can reproduce the input message by intercepting the ciphertext on the communication line. The following illustration explains this possible attack:

We have
$c_i = m_i$ XOR $Kc_i' = m_i$ XOR (LFSR-1 XOR Kc_i) = m_i XOR LFSR-1 XOR Kc_i;
Since control parameters are known, we can get the first k bytes of Kc, $(Kc_0, Kc_1, \ldots, Kc_i, \ldots, Kc_{k-1})$;
Since the initial state of the LFSR-1, Kc_i $(i = 0, \ldots, k-1)$ and c_i are known, we can get the first k bytes of m, $(m_0, m_1, \ldots, m_i, \ldots, m_{k-1})$;
Since we know the first k bytes of m, we can get the next k bytes of Kc, $(Kc_k, Kc_{k+1}, \ldots, Kc_{k+i}, \ldots, Kc_{2k-1})$.

From the aforementioned illustration, we demonstrate that an attacker can reproduce the original input message.

3.4.1. Assumption

The initial keystream *M* is either random or non-random, and control parameters, *M*, *N* and *cc*, are not securely protected.

3.4.2. Design

To remedy the "non-secure control parameters" problem, we propose the stream cipher *S*3. *S*3, as shown in Fig. 5, consists of the KG, two LFSRs, and three XOR operators. We choose two maximal-length LFSRs whose length are 33 and 37 bytes, which are relatively prime, respectively. Each LFSR is a primitive polynomial mod 2. The degrees of these two LFSRs are (33,13,0) and (37,6,4,1,0). The initial state

Table 3
Number of primitive operations of *S*1, *S*2, and *S*3

	XOR	INC	ADD	SHIFT
*S*1	2	3	2	0
*S*2	6	3	2	1
*S*3	7	3	2	1

(a byte string) of each LFSR is randomly generated with the exception of the rightmost byte. The rightmost byte is always set to 255 ($0 \times FF$) to avoid the case that the initial byte string is of all 0's. The stream cipher *S*3 is stated as follows:

Step 1. The LFSR-2 is clocked, if (Kc_i XOR m_i) mod 2 is equal to 1;
The LFSR-3 is clocked, if (Kc_i XOR m_i) mod 2 is equal to 0;
Step 2. The keystream, (Kc_i'), is obtained by using the XOR operation between the outputs of the LFSR-2 and the LFSR-3.
Step 3. The ciphertext, (c_i), is obtained by using the XOR operation between the input message, (m_i), and the keystream, (Kc_i').

3.4.3. Security analysis

The security of *S*3 relies on the randomness of the maximal-length LFSR. *S*3 includes two relatively primed LFSRs. The period of the combined LFSR is equal to the multiple of LFSR-2's period and LFSR-3's period. Therefore, *S*3 has a longer period than the "single LFSR" *S*2.

Although control parameters and the initial state might be known for both the LFSR-2 and the LFSR-3, *S*3 is still considered to be secure, since an attacker is not able to know which LFSR was clocked due to the secrecy of the input message, (m_i).

3.4.4. Efficiency analysis

Although *S*3 includes two LFSRs; however, for each input byte, only one LFSR is triggered. In essence, *S*3 requires only one more XOR operation than *S*2; therefore, *S*3 is as efficient as *S*2.

Table 4
Stream ciphers for GSM networks

Division of GSM base station	Proposed stream cipher
A	*S*1
B	*S*2
C	*S*3
D	*S*3

4. Stream ciphers for GSM networks

4.1. Comparative analysis of S1, S2, and S3

4.1.1. Security analysis

The level of security of *S*1, *S*2, and *S*3 is in an ascending order with *S*3 having the highest level of security. On a non-trusted machine; i.e. control parameters, *M*, *N* and *cc*, are not well protected, *S*3 is the only stream cipher which is immune from security attacks. On a trusted machine, both *S*2 and *S*3 are secure with the initial keystream *M* being non-random; and all three stream ciphers are secure with *M* being random.

4.1.2. Efficiency analysis

We evaluate the efficiency of *S*1, *S*2, and *S*3 in term of the number of primitive operations executed for a one-byte long input message. We have identified four primitive operations which are XOR, INC, ADD, and SHIFT. Table 3 details the number of primitive operations required by *S*1, *S*2, and *S*3.

By examining Table 3, we notice that *S*1 requires the least number of primitive operations; hence, it is the most efficient stream cipher among the three proposed stream ciphers. As for stream ciphers *S*2 and *S*3, their efficiency is almost the same, since *S*3 requires only one more XOR operation than *S*2.

4.2. Stream ciphers for GSM networks

Control parameters, *M*, *N* and *cc*, are stored in a base station (BS) of GSM networks. Only the BS can change their contents. Thus, the security of control parameters depends on the protection a BS can provide. The trusted computer system evaluation criteria (TCSEC) [2] defined by US Department of Defense (DoD) classify systems into four broad hierarchical divisions of enhanced security protection. They provide a basis for the evaluation of effectiveness of security controls built into automatic data processing system products. The criteria are divided into four divisions: D, C, B and A ordered in a hierarchical manner with the highest division (A) being reserved for systems providing the most comprehensive security. Each division represents a major improvement in the overall confidence one can place in the system for the protection of sensitive information. Divisions A, B, C and D provide verified protection, mandatory protection, discretionary protection, and minimal protection, respectively. For Division C or D BSs, we recommend that the stream cipher *S*3 be used. For Division B BSs, we suggest that the stream cipher *S*2 be considered, since the randomness of initial keystream *M* cannot be verified. For Division A BSs, we propose the stream cipher *S*1 to be used, since Division A BSs have the highest security requirement. Table 4 summarizes our recommendations.

5. Conclusions

5.1. Summary

In this paper, we focus on the security of transferring voice message on the GSM networks. A KG is presented. Three stream ciphers are designed in conjunction with the KG. Simulation results indicate that the KG can always produce key strings of evenly distributed 0's and 1's and with a very long period; i.e. it is very unlikely that repeated patterns will appear. Cryptanalysis and operational analysis show that the proposed stream ciphers are secure and efficient. These stream ciphers provide a comprehensive set of message encryption/decryption algorithms for GSM networks.

5.2. Future works

In addition to the message encryption and decryption algorithm (the A5 algorithm), entity authentication (the A3 algorithm) is another major concern of the security of GSM networks. However, the challenge–response based A3 is not very secure [4]. In our future works, we intend to enhance the A3 algorithm and propose a new authentication protocol.

References

[1] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, Wiley, New York, 1994.
[2] DoD 5200.28-STD, Department of Defense Trusted Computer System Evaluation Criteria, December 1985.
[3] D.G.W. Birch, I.J. Shaw, Mobile communications security-private or public, IEE June (1994).
[4] R. Bird, et al., Systematic design of a family if attack-resistant authentication protocols, IEEE Journal on Selected Areas in Communications 11 (5) (1993).
[5] C. Brookson, GSM security: a description of the reasons for security and the techniques, IEE (1994).
[6] J.C. Cooke, R.L. Brewster, Cryptographic security techniques for digital mobile telephones, Wireless Communications (1992).
[7] C. Kaufman, R. Perlamn, M. Speciner, Network Security: Private Communication in a Public World, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[8] Yi-Bing Lin, No wires attached, IEEE Potentials 14 (October) (1995).
[9] A. Mehrotra, L.S. Golding, Mobility and security management in the GSM system and some proposed future improvements, Proceedings of the IEEE 86 (7) (1998) 00.
[10] M. Mouly, M.-B. Pautet, GSM protocol architecture: radio subsystem signalling, IEEE 41st Vehicular Technology Conference, 1991.
[11] M. Mouly, M.-B. Pautet, The GSM system for mobile communications, 1992.
[12] J. Scourias, A brief overview of GSM, http://kbs.cs.tu-berlin.de/~jutta/gsm/js-intro.html, 1994.
[13] S.J. Shepherd, Public key stream ciphers, IEE (1994).
[14] K. Zeng, et al., Pseudorandom bit generators in stream-cipher cryptography, IEEE Computer (1991).