关于三个流密码的安全性*

张 斌[1+], 伍宏军[2], 冯登国[1], 鲍 丰[2]

[1](信息安全国家重点实验室(中国科学院 研究生院),北京 100049)

[2](Institute for Infocomm Research, 119613, Singapore)

On the Security of Three Stream Ciphers

ZHANG Bin[1+], WU Hong-Jun[2], FENG Deng-Guo[1], BAO Feng[2]

[1](State Key Laboratory of Information Security (Graduate School, The Chinese Academy of Sciences), Beijing 100049, China)

[2](Institute for Infocomm Research, 119613, Singapore)

+ Corresponding author: Phn: +86-10-88258713, Fax: +86-10-88258713, E-mail: mzb_123@hotmail.com, http://www.is.ac.cn

Received 2004-02-13; Accepted 2004-10-09

Zhang B, Wu HJ, Feng DG, Bao F. On the security of three stream ciphers. ***Journal of Software*, 2005,16(7): 1344−1351.** DOI: 10.1360/jos161344

Abstract: In this paper three newly proposed stream ciphers S1, S2 and S3 are analyzed. These stream ciphers are designed with respect to different levels of GSM security. The results show that both S1 and S2 are vulnerable to the known plaintext attacks and S3 can not decrypt correctly. With negligible amount of computation and few known keystream bytes, S1 and S2 can be broken completely. Furthermore, simulation results show that S3 cannot work correctly. The conclusion is that these stream ciphers are either extremely weak or poorly designed so that they cannot play the role as the designers hope in GSM network security.

Key words: stream cipher; security; GSM network; linear feedback shift register; bytes

摘 要: 对 3 个新近提出的流密码 S1,S2 及 S3 进行了分析.这 3 个流密码被设计用于 GSM 网络加密,且分别对应于不同的安全性等级.结果表明,S1 和 S2 都易受已知明文攻击,而 S3 不能正确解密.只需少量的密文字节和可以忽略的计算量就能够完全破解 S1 和 S2.模拟实验结果表明,S3 不能正确工作.结论是这 3 个流密码要么及其脆弱,要么就是不能正确解密,因此它们并不能在 GSM 网络安全方面扮演设计者所希望的角色.

关键词: 流密码;安全性; GSM 网络;线性反馈移位寄存器;字节

中图法分类号: TP309 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60273027 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.G1999035802 (国家重点基础研究发展规划(973)); the National Science Fund of China for Distinguished Young Scholars under Grant No.60025205 (国家杰出青年学者科学基金)

ZHANG Bin was born in 1976. He is a Ph.D. candidate at the State Key Laboratory of Information Security, Graduate School of the Chinese Academy of Sciences. His current research interests include analysis and design of stream ciphers. **WU Hong-Jun** was born in 1974. He is an Associate Scientist at the Institute for InfoComm Research, Singapore. His researches areas are cryptography and information security. **FENG Deng-Guo** was born in 1965. He is a professor and doctoral supervisor at the State Key Laboratory of Information Security, Graduate School of the Chinese Academy of Sciences. His research areas are information security and network security. **BAO Feng** was born in 1962. He is a Leading Scientist at the Institute for InfoComm Research, Singapore. His research areas include cryptography, distributed computing and automata theory.

1 Introduction

Mobile communication has become an indispensable part of ordinary life in today's world. People can communicate with each other anywhere and anytime through mobile phones. However, the openness of wireless communications has caused many security problems. It is now believed that the security service is essential to the success of a mobile communication network. The Global System for Mobile Communications (GSM)[1,2] is the standard for digital mobile communications. It involves a set of security features such as applying stream cipher A5 in encryption/decryption.

Due to the fact that the A5 algorithm recommended in GSM standard is a proprietary algorithm and A5/1 is subject to export control, many efforts have been made towards designing new non-proprietary encryption algorithms. In Refs.[3,4], three stream ciphers are proposed for GSM applications. So far, we do not see any analysis of these three stream ciphers elsewhere. They are claimed to be efficient and secure. However, we will show in this paper that the three stream ciphers proposed are either extremely weak or poorly designed. Both S1 and S2 are vulnerable to known plaintext attacks and S3 cannot decrypt correctly. With negligible amount of computation and few known keystream bytes, S1 and S2 can be broken completely.

The organization of this paper is as follows: first we will give a review on the three newly proposed stream ciphers in section 2, especially on the key generator and two operation modes of stream cipher S2. Our analysis is presented in section 3 including the theoretical complexities to break S1, S2 and the experimental results illustrating the decryption failure of S3. Finally, some conclusions are given in section 4.

2 Three Newly Proposed Stream Ciphers

GSM is the first mobile communication system that has comprehensive security features. The A5 algorithm is a proprietary algorithm used in GSM for message encryption/decryption. Since A5/1 algorithm is subject to more and more suspicions on its security[5,6], many efforts have been made to design new stream ciphers for GSM network. In Refs.[3,4] the authors proposed three simple stream ciphers S1, S2, S3 to substitute the A5 algorithm with respect to different levels of security. Unfortunately, as we will show below, the three stream ciphers are all poorly designed.

We will give a complete description of the three stream ciphers S1, S2 and S3 in the following. The three stream ciphers are all based on the following key generator (KG).

2.1 Key generator (KG)

For an initial keystream of length l bytes, $(M_0,M_1,\ldots,M_{l-1})$, and an input message of length n bytes, $(m_0,m_1,\ldots,m_{n-1})$, the keystream $(K_0,K_1,\ldots,K_{n-1})$ of length n bytes, is generated according to the following procedures:

```
Step 1. Let $i=0, j=0, N=170, cc=0$;
Step 2. Compute $M_j=(M_j+cc)\bmod 2^8$; If $j=0$ then $M_j=M_j\oplus N$; else $M_j=M_j\oplus M_{j-1}$;
Step 3. Let $K_i=M_j$; $N=(N+m_i)\bmod 2^8$; $i=i+1$; $j=j+1$; $cc=cc+1$;
        If $j=l$ then reset $j$ to 0;
        If $cc=256$ then reset $cc$ to 0;
        If $i=n$ then exit; else goto Step 2,
```

where N is an eight-bit string used to change M_0 by XOR operation at the beginning of each cycle; $\oplus$ denotes bit-wise XOR. cc is a number added to M_j $(j=0,\ldots,l-1)$ so as to increase the randomness of M_j. Note that the initial value of cc and N can be any number from 0 to 255. In the above design, take the number 170 $(10101010)_2$ as the default value of N and 0 as the default value of cc.

2.2 Stream cipher S1

The stream cipher S1, as shown in Fig.1, consists of the KG and an XOR operator. The ciphertext (c_i) is obtained by XOR between the input message (m_i) and the output (K_i) of KG on a byte-by-byte basis.

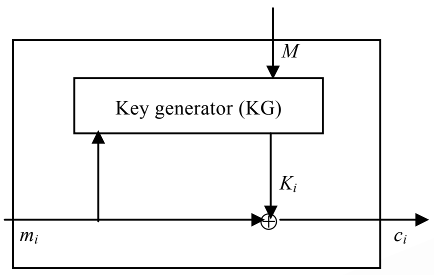

Fig.1 The stream cipher S1

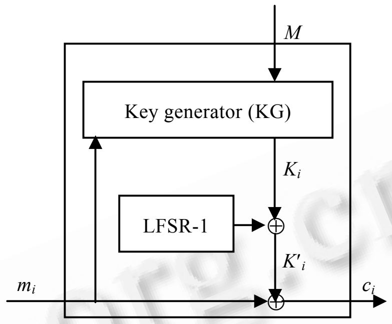

Fig.2 The stream cipher S2

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

2.3 Stream cipher S2

To increase the randomness of the generated keystream of key generator (KG), in Ref.[3] the authors proposed the following stream cipher S2, as shown in Fig.2. It consists of the KG, the LFSR-1, and two XOR operators, where the LFSR-1 is a maximum-length LFSR with tap sequence (32,7,6,2,0).

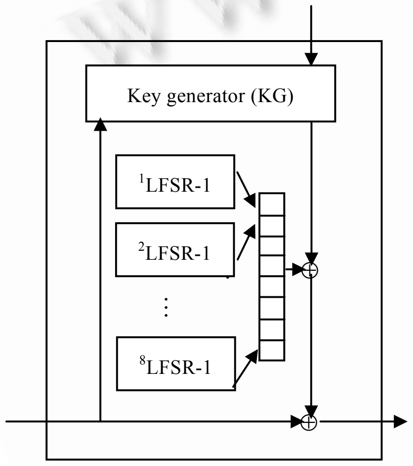

Fig.3 Mode 1 of stream cipher S2

The keystream K'_i is obtained by using XOR operation between the output of KG and the output stream of LFSR-1. The ciphertext (c_i) is the outcome of the XOR operation between the message bytes (m_i) and the keystream bytes (K'_i). The keys of stream cipher S2 are the initial state of LFSR-1 and the secret parameters of KG.

In general, there are two operation modes of stream cipher S2, Fig.3 shows the mode 1.

Mode 2: instead of using eight parallel LFSR-1s as in mode 1, this mode is just the direct implementation of S2 with the clock of LFSR-1 eight times as fast as that of KG.

2.4 Stream cipher S3

This cipher is designed as the most reliable stream cipher for GSM applications. It consists of the KG, two LFSRs, and three XOR operators. The two LFSRs are maximum-length LFSRs with known tap sequences (33,13,0) and (37,6,4,1,0) respectively. As Fig.4 shows, this cipher works according to the following rule:

Step 1. If ($K_i \oplus m_i$)mod2 is 1, then the LFSR-2 is clocked;
Else ($K_i \oplus m_i$)mod2=0, then the LFSR-3 is clocked;

Step 2. The keystream K'_i is obtained by using XOR operation between the output of the LFSR-2 and the output of LFSR-3;

Step 3. The ciphertext, (c_i), is obtained by using the XOR operation between the input message, (m_i), and the keystream (K'_i) on a byte-by-byte basis.

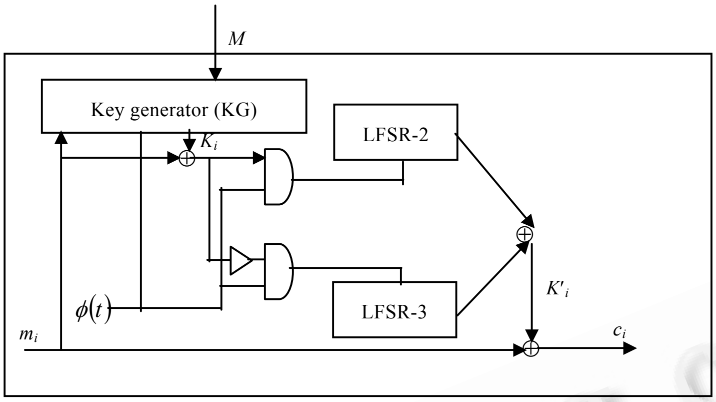

Fig.4 The stream cipher S3

The keys of stream cipher S3 are the initial states of the LFSR-2, LFSR-3, and the secret parameters of the KG. The security level of the stream ciphers S1, S2, S3 is in an ascending order with S3 having the highest level of security. In Ref.[3], the authors conclude that if the initial keystream M being random, then all the three stream ciphers are secure. However, as we will show below that it is not true.

3 Cryptanalysis of the Three Stream Ciphers S1, S2, S3

In this section, we will break the stream ciphers S1, S2 and show that the stream cipher S3 can not correctly decrypt. Since known-plaintext attack is the most basic attack that a good stream cipher should resist successfully, we will simply implement this kind of attack on these three stream ciphers.

3.1 A known-plaintext attack on S1

Note that S1 is the direct implementation of the key generator to encrypt messages and there are no complicated nonlinear permutations used in this cipher. This is S1's vital flaw. Assume that the key of S1 consists of the initial keystream $(M_0,M_1,\ldots,M_{l-1})$ and the secret parameters N and cc. We know a segment of the generated keystream (K_i). Our task is to recover the key from the known segment of (K_i). Note that we process on a byte-by-byte basis.

We have following equations according to the description of KG:

$$\begin{cases} m_0 \oplus c_0 = K_0 = (M_0 + cc) \oplus N \\ m_1 \oplus c_1 = K_1 = (M_1 + cc + 1) \oplus K_0 \\ \quad \vdots \\ m_{l-1} \oplus c_{l-1} = K_{l-1} = (M_{l-1} + cc + l - 1) \oplus K_{l-2} \\ m_l \oplus c_l = K_l = (K_0 + cc + l) \oplus (N + m_0 + \ldots + m_{l-1}) \\ m_{l+1} \oplus c_{l+1} = K_{l+1} = (K_1 + cc + l + 1) \oplus K_l \end{cases} \tag{1}$$

where ⊕ denotes bit-wise XOR and + denotes mod2^8 addition. m_i and c_i denote the ith bytes of the messages and ciphertexts, respectively. From (1) we get:

$$\begin{cases} m_i \oplus c_i \oplus m_{i+1} \oplus c_{i+1} = (M_{i+1} + cc + i + 1), 0 \le i \le l-2 \\ m_l \oplus c_l = K_l = (K_0 + cc + l) \oplus (N + m_0 + \ldots + m_{l-1}) \\ m_{l+1} \oplus c_{l+1} = K_{l+1} = (K_1 + cc + l + 1) \oplus K_l \end{cases} \tag{2}$$

Thus we can recover cc from the last equation of (2) by an exhaustive search through $\{0,1\}^8$ which is computationally negligible. Then from the second last equation of (2) we can also restore N by an exhaustive over

$\{0,1\}^8$. The computation amount is also negligible. With the knowledge of N and cc, we can simply recover the initial keystream $(M_0,M_1,\ldots,M_{l-1})$ using l exhaustive searches through $\{0,1\}^8$, thus the total complexity of above procedures is $O(l)$ with l+2 bytes of keystream known, which means the stream cipher S1 is extremely weak.

3.2 An known-plaintext attack on S2

We first focus on the operation mode 1 of S2. Assume that the ^{1}LFSR-1 corresponds to the least significant bit of the output byte. We know a segment of the generated keystream and the feedback polynomial of LFSR-1. Our aim is to recover the initial states of the eight LFSRs and the initial keystream M, secret parameters N and cc.

Our basic technique is that when adding $(x+y)\bmod 2^8$, it has the same effect as $x^0\oplus y^0$ with respect to the least significant bits x^0 and y^0, where $\oplus$ denotes the XOR operation. Thus we have:

$$\begin{cases} m_0^0\oplus c_0^0=K_0^0\oplus x_1^0=M_0^0\oplus cc^0\oplus N^0\oplus x_1^0 \\ m_1^0\oplus c_1^0=K_1^0\oplus x_1^1=M_1^0\oplus cc^0\oplus 1\oplus K_0^0\oplus x_1^1 \\ \qquad\vdots \\ m_{l-1}^0\oplus c_{l-1}^0=K_{l-1}^0\oplus x_1^{(l-1)}=M_{l-1}^0\oplus cc^0\oplus (l-1)_2\oplus K_{l-2}^0\oplus x_1^{(l-1)} \\ m_l^0\oplus c_l^0=K_l^0\oplus x_1^l=K_0^0\oplus cc^0\oplus N^0\oplus m_0^0\oplus\cdots\oplus m_{l-1}^0\oplus x_1^l\oplus (l)_2 \\ m_{l+1}^0\oplus c_{l+1}^0=K_{l+1}^0\oplus x_1^{(l+1)}=K_1^0\oplus cc^0\oplus (l+1)_2\oplus K_0^0\oplus x_1^{(l+1)} \\ \qquad\vdots \\ m_{2l-1}^0\oplus c_{2l-1}^0=K_{2l-1}^0\oplus x_1^{(2l-1)}=K_{l-1}^0\oplus cc^0\oplus 1\oplus K_{l-2}^0\oplus x_1^{2l-1} \end{cases} \tag{3}$$

with respect to ^{1}LFSR-1, where $x^i{}_1$ denotes the ith output bit of ^{1}LFSR-1 and $(\bullet)_2$ denotes the least significant bit of the argument's binary representation. Note that here $m_i^0, c_i^0, K_i^0, N^0, cc^0$ denote the least significant bits of corresponding bytes. Matching the ith and the $(i+l)$th equation of (3) together for $0\le i\le l-1$, we have:

$$\begin{cases} m_0^0\oplus c_0^0\oplus m_l^0\oplus c_l^0=x_1^0\oplus x_1^l\oplus cc^0\oplus (l)_2\oplus N^0\oplus m_0^0\oplus\ldots\oplus m_{l-1}^0 \\ m_1^0\oplus c_1^0\oplus m_{l+1}^0\oplus c_{l+1}^0=x_1^1\oplus cc^0\oplus (l+1)_2\oplus \left(m_0^0\oplus c_0^0\oplus x_1^0\right)\oplus x_1^{l+1} \\ \qquad\vdots \\ m_{l-1}^0\oplus c_{l-1}^0\oplus m_{2l-1}^0\oplus c_{2l-1}^0=x_1^{l-1}\oplus cc^0\oplus (2l-1)_2\oplus \left(m_{l-2}^0\oplus c_{l-2}^0\oplus x_1^{l-2}\right)\oplus x_1^{2l-1} \end{cases} \tag{4}$$

There are altogether four combinations for the value of $\left(cc^0,N^0\right)$. We can try every combination, for it actually has no effect on the total complexity of our attack. So we can safely deal with the situation that $\left(cc^0,N^0\right)$ is known. In this case, we can transform the variables x_1^i with $i\ge 32$ in (4) into linear combinations of $\left(x_1^0,x_1^1,\ldots,x_1^{31}\right)$. For GSM applications where $l>>32$, we can get 32 linearly independent equations from above equation system with overwhelming probability. Then we solve these linearly independent equations to recover the initial state $\left(x_1^0,x_1^1,\ldots,x_1^{31}\right)$ of ^{1}LFSR-1. With the knowledge of $\left(x_1^0,x_1^1,\ldots,x_1^{31}\right)$, we can easily recover the least significant bits $\left(M_0^0,M_1^0,\ldots,M_{l-1}^0\right)$ according to (3).

So far, we have restored the following initial values $\left(x_1^0,x_1^1,\ldots,x_1^{31}\right)$, $\left(cc^0,N^0\right)$ and $\left(M_0^0,M_1^0,\ldots,M_{l-1}^0\right)$. Note that with the knowledge of the least significant bits, we can get the carry to forward bits when adding $(x+y)\bmod 2^8$. Thus we can get similar equation systems as (3) with respect to the second least significant bits. Assume we have got the ith least significant bits and the initial state of ${}^{i+1}$LFSR-1, then with respect to the i+1th least significant bits and the initial state of ${}^{i+2}$LFSR-1 with $0\le i\le 6$ we have:

$$\begin{cases} m_0^{i+1} \oplus c_0^{i+1} = K_0^{i+1} \oplus x_{i+2}^0 = M_0^{i+1} \oplus cc^{i+1} \oplus A_0 \oplus N^{i+1} \oplus x_{i+2}^0 \\ m_1^{i+1} \oplus c_1^{i+1} = K_1^{i+1} \oplus x_{i+2}^1 = M_1^{i+1} \oplus cc^{i+1} \oplus A_1 \oplus K_0^{i+1} \oplus x_{i+2}^1 \\ \vdots \\ m_{l-1}^{i+1} \oplus c_{l-1}^{i+1} = K_{l-1}^{i+1} \oplus x_{i+2}^{l-1} = M_{l-1}^{i+1} \oplus cc^{i+1} \oplus A_{l-1} \oplus K_{l-2}^{i+1} \oplus x_{i+2}^{l-1} \\ m_l^{i+1} \oplus c_l^{i+1} = K_l^{i+1} \oplus x_{i+2}^l = K_0^{i+1} \oplus cc^{i+1} \oplus A_l \oplus N^{i+1} \oplus B \oplus x_{i+2}^l \\ m_{l+1}^{i+1} \oplus c_{l+1}^{i+1} = K_{l+1}^{i+1} \oplus x_{i+2}^{l+1} = K_1^{i+1} \oplus cc^{i+1} \oplus A_{l+1} \oplus K_0^{i+1} \oplus x_{i+2}^{l+1} \\ \vdots \\ m_{2l-1}^{i+1} \oplus c_{2l-1}^{i+1} = K_{l-1}^{i+1} \oplus x_{i+2}^{2l-1} = K_{l-1}^{i+1} \oplus cc^{i+1} \oplus A_{2l-1} \oplus K_{l-2}^{i+1} \oplus x_{i+2}^{2l-1} \end{cases} \tag{5}$$

where A_i and B denote the carriers such that: for $0 \le j \le l-1$, A_j is the carry to the i+1th bit calculated from $\left(j \bmod 2^8\right)_2 + \left(M_j^i, M_j^{i-1}, \ldots, M_j^0\right)_2 + \left(cc^i, cc^{i-1}, \ldots, cc^0\right)_2$; for $l \le j \le 2l-1$, A_j is the carry to the i+1th bit calculated from

$$\left(j \bmod 2^8\right)_2 + \left(K_{j-l}^i, K_{j-l}^{i-1}, \ldots, K_{j-l}^0\right)_2 + \left(cc^i, cc^{i-1}, \ldots, cc^0\right)_2 \tag{6}$$

B is the carry to the i+1th bit calculated from

$$\left(N^i, N^{i-1}, \ldots, N^0\right)_2 + \sum_{k=1}^{l-1}\left(m_k^i, m_k^{i-1}, \ldots, m_k^0\right)_2 \tag{7}$$

Note that all the A_js and B can be easily calculated from the known information. Thus we will regard them as known parameters. Other parameters such as K_j^{i+1}, m_j^{i+1} and c_j^{i+1} in (5) denote the i+1th least significant bits of the corresponding jth bytes. x_{i+2}^j denotes the jth output bit by the ${}^{i+2}$LFSR-1. Matching the ith and the $(i+l)$th equations of (5) together for $0 \le i \le l-1$, we have:

$$\begin{cases} m_0^{i+1} \oplus c_0^{i+1} \oplus m_l^{i+1} \oplus c_l^{i+1} = x_{i+2}^0 \oplus x_{i+2}^l \oplus cc^{i+1} \oplus A_l \oplus B \oplus N^{i+1} \\ m_1^{i+1} \oplus c_1^{i+1} \oplus m_{l+1}^{i+1} \oplus c_{l+1}^{i+1} = x_{i+2}^1 \oplus cc^{i+1} \oplus A_{l+1} \oplus \left(m_0^{i+1} \oplus c_0^{i+1} \oplus x_{i+2}^0\right) \oplus x_{i+2}^{l+1} \\ \vdots \\ m_{l-1}^{i+1} \oplus c_{l-1}^{i+1} \oplus m_{2l-1}^{i+1} \oplus c_{2l-1}^{i+1} = x_{i+2}^{l-1} \oplus cc^{i+1} \oplus A_{2l-1} \oplus \left(m_{l-2}^{i+1} \oplus c_{l-2}^{i+1} \oplus x_{i+2}^{l-2}\right) \oplus x_{i+2}^{2l-1} \end{cases} \tag{8}$$

Because there are only four combinations of the value $\left(cc^{i+1}, N^{i+1}\right)$, we can simply try every combination which has actually no effect on the total complexity of our attack. So we can also deal with the situation that $\left(cc^{i+1}, N^{i+1}\right)$ is known. In this case, we can transform the variables x_{i+2}^j with $j \ge 32$ in (8) into linear combinations of $\left(x_{i+2}^0, x_{i+2}^1, \ldots, x_{i+2}^{31}\right)$. For GSM applications where $l >> 32$, we can get 32 linearly independent equations from above equation system with overwhelming probability. Then we solve these linearly independent equations to recover the initial state $\left(x_{i+2}^0, x_{i+2}^1, \ldots, x_{i+2}^{31}\right)$ of ${}^{i+2}$LFSR-1. With the knowledge of $\left(x_{i+2}^0, x_{i+2}^1, \ldots, x_{i+2}^{31}\right)$, we can easily recover the least significant bits $\left(M_0^{i+1}, M_1^{i+1}, \ldots, M_{l-1}^{i+1}\right)$ according to (5). The following is our total attack:

```
Parameters: feedback polynomial with tap sequence (32,7,6,2,0);
Step 1. input the value of $l$;
    Derive a linear equation system of 2$l$ linear equations from the first 2$l$ bytes of keystream as in (3)
    and solve it to get the initial values $\left(x_1^0, x_1^1, \ldots, x_1^{31}\right)$, $\left(cc^0, N^0\right)$ and $\left(M_0^0, M_1^0, \ldots, M_{l-1}^0\right)$;
Step 2. Compute $A_j$ and $B$, for $0 \le j \le 2l-1$, from the known bits;
    for $i$=0 to 6
    Derive a linear equation system of 2$l$ linear equations from the first 2$l$ bytes of keystream as in (5);
    Derive a linear equation system of 32 linear equations on the initial state $\left(x_{i+2}^0, x_{i+2}^1, \ldots, x_{i+2}^{31}\right)$ and
```

guess the value of $\left(cc^{i+1}, N^{i+1}\right)$ to solve this system;

Deduce $\left(M_0^{i+1}, M_1^{i+1}, \ldots, M_{l-1}^{i+1}\right)$.

Since the complexity of solving a system of 32 linear equations is only constant magnitude, the total complexity of above attack is *O*(1) with 2*l* known keystream bytes, which means the operation mode 1 of stream cipher S2 is extremely weak too.

As for the operation mode 2 of stream cipher S2, it is simply to substitute the variables x_1^i for $x_1^{8\cdot i}$ in the equation system (3) to determine the least significant bits of $\left(M_0, M_1, \ldots, M_{l-1}\right)$ as follows:

$$\begin{cases} m_0^0 \oplus c_0^0 = K_0^0 \oplus x_1^0 = M_0^0 \oplus cc^0 \oplus N^0 \oplus x_1^0 \\ m_1^0 \oplus c_1^0 = K_1^0 \oplus x_1^{8\cdot 1} = M_1^0 \oplus cc^0 \oplus 1 \oplus K_0^0 \oplus x_1^{8\cdot 1} \\ \qquad\qquad\qquad\qquad \vdots \\ m_{l-1}^0 \oplus c_{l-1}^0 = K_{l-1}^0 \oplus x_1^{8(l-1)} = M_{l-1}^0 \oplus cc^0 \oplus (l-1)_2 \oplus K_{l-2}^0 \oplus x_1^{8(l-1)} \\ m_l^0 \oplus c_l^0 = K_l^0 \oplus x_1^{8l} = K_0^0 \oplus cc^0 \oplus N^0 \oplus m_0^0 \oplus \cdots \oplus m_{l-1}^0 \oplus x_1^{8l} \oplus (l)_2 \\ m_{l+1}^0 \oplus c_{l+1}^0 = K_{l+1}^0 \oplus x_1^{8(l+1)} = K_1^0 \oplus cc^0 \oplus (l+1)_2 \oplus K_0^0 \oplus x_1^{8(l+1)} \\ \qquad\qquad\qquad\qquad \vdots \\ m_{2l-1}^0 \oplus c_{2l-1}^0 = K_{2l-1}^0 \oplus x_1^{8(2l-1)} = K_{l-1}^0 \oplus cc^0 \oplus 1 \oplus K_{l-2}^0 \oplus x_1^{8(2l-1)} \end{cases} \tag{9}$$

The procedures for solving (9) are the same as those for solving (3) of mode 1, so we omit them. When determining the *i*+1th least significant bits with the knowledge of *j*th $(j \le i)$ least significant bits, we simply substitute the variables x_{i+2}^j for x_1^{j+8k} with $0 \le k \le 2l-1$. Other procedures are the same as those in mode 1, we omit them too. The complexity of our attack on mode 2 is also *O*(1) with 2*l* known keystream bytes, thus we can safely affirm that the stream cipher S2 is so weak in security that it should not be used in GSM applications.

3.3 Analysis of the stream cipher S3

In this section, we will show that the proposed stream cipher S3 can not decrypt correctly. From Fig.4, we can see that the two LFSRs work in a stop/go manner whose clocks are controlled by the XOR outcome of K_i and m_i. Since S3 works in a self-synchronization fashion, it is obvious that if the same initial states and the same initial keystream are enclosed into the two LFSRs and the key generator respectively, then the clock behaviours of LFSR-2 and LFSR-3 on the receiver side are not the same as those on the sender side. Thus in general the output of the XOR outcome K'_i on the receiver side is not the byte encrypted to the message byte m_i, which means S3 cannot decrypt correctly. On the other side, if one wish to choose the initial states of the two LFSRs and the initial keystream of the key generator in such a way that on the receiver side it can decrypt correctly, then he will be confronted with the problem that without the knowledge of the message m_i, he has to simulate the correct clock behaviour that can decrypt the ciphertext c_i, which is impossible for practical applications.

Experimental results. We have made simulation experiments to test the stream cipher S3 in C language on a Pentium 4 2.5GHz processor. All the experimental results show that S3 cannot decrypt correctly. Here we only list the results of mode 1 which apply eight parallel LFSRs for LFSR-2 and LFSR-3 respectively. The tap sequences of the two LFSRs are (33,13,0) and (37,6,4,1,0) respectively. We use RC4 as random noise source to provide the initial states of the LFSRs and the initial keystream of KG. The results are shown in Tables 1, 2, 3. In each table, the bytes order is assumed as from left to right and from top to bottom. Similarly, we can use LFSRs whose clocks are eight times as fast as that of KG, the results are the same, we omit them.

Table 1 The 100 input message bytes

b0	aa	f7	3c	af	b8	ca	ae	f6	57
2b	dd	17	6a	fc	45	15	c4	69	7f
a5	59	20	13	39	19	12	06	2e	23
8f	75	3d	16	a8	c1	5b	3b	d4	14
97	30	dd	08	73	80	c6	6a	35	54
1a	ce	ba	cc	aa	a1	35	f0	c9	42
0e	1e	9b	f9	0b	2e	20	34	ac	0c
1c	4a	18	54	1e	90	34	d9	33	ff
bf	23	67	cf	be	97	ed	c5	f0	80
d1	20	65	55	18	b4	e1	11	72	e2

Table 2 The 100 ciphertext bytes

86	6d	6e	0f	dd	3b	bb	20	3b	5e
58	8d	d2	94	f8	6f	74	5e	92	35
41	f2	16	cc	ad	7a	2e	88	cc	a3
82	26	37	54	01	73	5c	0c	cc	b9
0f	d8	b6	4f	b9	1b	b5	4f	95	e4
60	3b	fe	f2	8a	c6	74	64	76	80
93	df	a3	fe	5e	4a	9f	fe	ad	a5
0a	3d	cc	af	63	35	88	c1	a4	94
6b	f1	e2	c2	64	25	c4	93	a2	f3
f9	ed	28	40	11	0c	2f	8f	ba	36

Table 3 The 100 decrypted bytes

b0	05	f7	64	1c	05	c4	25	ab	af
58	ce	55	30	67	0a	a0	a4	23	19
96	cc	63	17	81	19	12	06	2e	23
5b	b7	bd	6b	0e	53	f1	ff	8a	14
97	30	b6	19	6c	89	0b	7c	f7	03
97	ea	e5	66	af	99	35	f0	9f	90
d1	b3	a1	b5	3e	e1	7c	42	9e	ca
a5	ad	87	91	96	df	bb	39	80	59
4c	e8	84	fc	5a	5b	28	36	8e	3f
37	69	0a	47	ae	64	97	bf	0a	ac

We can see from above tables that the decrypted bytes are hardly the same as those in the corresponding input messages, which illustrates that the stream cipher S3 cannot decrypt correctly in general.

4 Conclusions

In this paper, we analyze three stream ciphers S1, S2 and S3 designed for GSM security applications. Our results show that both S1 and S2 are vulnerable to the known-plaintext attacks and S3 cannot decrypt correctly. Hence, the three stream ciphers should not be used in practice. It is an interesting problem and our future work to improve these three stream ciphers against the attacks in this paper.

References:

[1] Mouly M, Pautet MB. GSM protocol architecture: Radio subsystem signaling. In: Proc. of the 41st IEEE Vehicular Technology Conf. 1991.

[2] Scourias J. A brief overview of GSM. 1994. http://kbs.cs.tu-berlin.de/~jutta/gsm/js-intro.html

[3] Lo CC, Chen YJ. Stream ciphers for GSM networks. Computer Communications, 2001,24(11):1090−1096.

[4] Lo CC, Chen YJ. Secure communication mechanisms for GSM networks. IEEE Trans. on Consumer Electronics, 1999,45(4):1074−1080.

[5] Biryukov A, Shamir A, Wagner D. Real time cryptanalysis of A5/1 on a PC. In: Schneier B, ed. Fast Software Encryption 2000. LNCS1978, New York: Springer-Verlag, 2001. 1−18.

[6] Biham E, Dunkelman O. Cryptanalysis of the A5/1 GSM stream cipher. In: Roy B, Okamoto E, eds. Progress in Cryptology-INDOCRYPT 2000. LNCS1977, 2000. 43−51.