A5 Encryption In GSM

Oliver Damsgaard Jensen
Kristoffer Alvern Andersen

June 2017

Abstract

A5 is a family of symmetric stream ciphers most famously used as the encryption schemes in GSM [1] and succeeding technologies. The A5 algorithms are designed for simple commodity hardware with focus on security and speed. The short key length used in A5, along with other vulnerabilities, makes GSM prone to attacks. The architecture and implementation of the algorithms are also flawed and can be abused from not only governments, but adversaries without extensive computational power. The decryption can be done in close to real time. This paper will present the encryption scheme used in the 2G GSM network, the A5/1 algorithm, as well as some of the associated vulnerabilities.

1 Introduction

The main integrity objective of cryptographic schemes is to make private data incomprehensible for adversaries. However, the privacy of data sent through the GSM network is close to non-existing. Any malicious eavesdropper can decrypt GSM packets in real time with commodity hardware[8]. Thus, privacy is threatened by any adversary with a standard computer. The cellular telecommunication infrastructure is mainly built for GSM based technologies, which leads to great potential for widespread attacks. Though there exist several implementations of the A5 algorithm, the most commonly used are A5/0, A5/1, and A5/2. The version implemented in the network is often dependent on geographical location. A5/1 is used in both Europe, and the US, and is the strongest of the three. A5/2, which is a deliberate weakening of the A5/1 algorithm, is used in certain export regions, such as Asia. The non-encrypted version,the A5/0 algorithm, is used in countries under UN sanctions, and certain third world countries[6]. We will focus on the stronger of the three, but still weak A5/1 algorithm, and highlight its properties and flaws.

2 GSM

GSM (Global System for Mobile Commmunications) is an ETSI[2] standard describing protocols for 2G digital cellular mobile networksGSM was the first near to global standard of wireless telecommunication. With over 90 % market share, GSM based technologies like EDGE, UMTS, HSPA and recent LTE-Advanced provides wireless communication services for more than 6 billion people worldwide[1]. Information security is an important issue in the wireless mobile technologies nowadays. However, When the GSM standard was designed from 1982-1991, the level of security specifications regarding both authentication and encryption were limited. Specific algorithms were never officially published. If this was an attempt to gain security through obscurity is not known, however this practice is not recommended by NIST[2]. The Algorithms were reverse-engineered (cite anderson ross) and some were leaked, leading to revelations of several possible attacks. Various versions of A3, A5 and A8 are used in GSM to provide Authentication and Integrity.

The default security settings in GSM were standardized and assigned to individual geographical preferences. Given the "in air availability" of signals, new opportunities arose for cleaver eavesdroppers. Within a few months after the release, most of the cryptographic schemes had been compromised and some were even proven to be close to useless.

3 A5 - Data Integrity for GSM

GSM was originally designed for basic commodity phones and a security scheme could therefore only enforce simple hardware requirements. A5 was specified to encrypt over-the-air transmissions on the GSM network, and is a symmetric stream cipher[3]. The algorithm takes 228 bits of plain text as input and outputs 228 bits of cipher text. Each block of 228 bits is called a "frame", where the first 114 bits represents data sent from unit

[1] GSM - Global System for Mobile communication
[2] ETSI - European Telecommunications Standards Institute is an independent, not-for-profit, standardization organization
[3] Symmetric stream cipher - Symmetric in that the encryption and decryption key is the same, stream cipher implies that encryption is done continuously and not in bulks

A to unit B, and the last 114 bits are data received by unit A from unit B. Each frame has a duration of 4.615 ms, allowing 28̂ frames to be sent every second[8].

Over time, several A5-versions have been developed, but they all share the same main idea. An A5 algorithm takes the session key K_c (symmetric) and a frame counter F_n, and generates 228 pseudo random bits (PRAND), called a key stream. The key stream is then XORed with a 228 bit segment of plain text, yielding 228 bits of ciphertext. Figure 1 shows a schema of the A5 data flow.

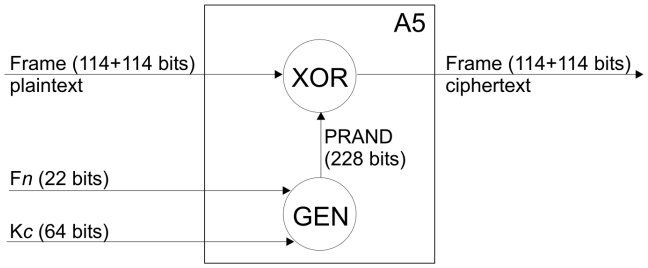

Figure 1: A5 schema

The encryption itself is just a simple XOR operation for each bit, which may seem very naive. If the encryption itself is simple, what makes A5 secure? In A5, it is the generation of pseudo random bits (function GEN in figure 1) that is important. The different A5-versions offer different levels of security by implement GEN differently.

3.1 A5/0 - Unencrypted Stream Cipher

A5/0 is the weakest of the A5 versions as it does not offer any encryption at all. It is a no-operation cipher, that generates the pseudo random bits by negating the input frame, thus leaving out the XOR function. The result is an algorithm that outputs the plain text it received as an input. This version is found in third world countries and countries with UN sanctions[6].

3.2 A5/1 - Strong Encryption

A5/1 is the strongest and most popular of the GSM A5 versions and it is used in both Europe and in the US. A5/1's pseudo random bits are generated by 3 linear feedback shift registers (LFSRs). An LFSR is a shift register whose input is a linear function of its previous state. The register's state is decided by several tap-bits, and the linear feedback function is an XOR of all the register's tap bits.

When a register is clocked, its tapped bits are XORed and the result is stored in the registers least significant bit (LSB). The registers most significant bit is shifted out of the register, and its value is forgotten. Clocking a register is a fundamental operation for this algorithm, and the term will be used excessively throughout this paper. Figure 2 shows one cycle of a register clocking, where register index number 13 and 17 holds the value of the tapped bits (index 13 and 17 are arbitrarily chosen for the sake of the example).

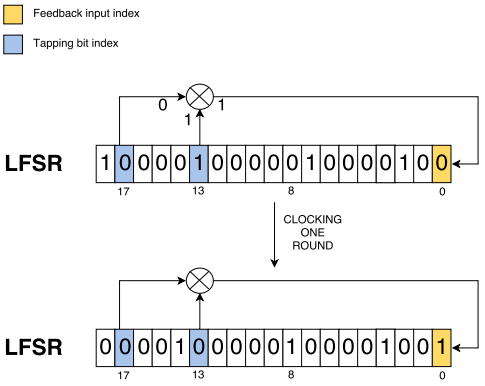

Figure 2: A5 schema

A5/1 consist of 6 steps, where the first 5 steps generate the pseudo random key stream, and the last step applies the key stream to the plain-text, thereby generating the cipher text.

3.2.1 Step 1 - Initializing the LFSRs

As mentioned, A5/1 uses 3 LFSRs to generate its pseudo random key stream. Before they can generate pseudo random numbers, they have to be initialized and preprocessed. Furthermore, they are all unique, and have different lengths, and different indices that are tapped when being clocked. In this initializing step, all the registers are filled with zero-values.

LFSR number	Length in bits	Clocking bit	Tapped bits
1	19	8	13,16,17,18
2	22	10	20,21
3	23	10	7,20,21,22

Figure 3: LFSR infomation table

Figure 3 shows the specification of each LFSR. The clocking bits will come into play later, in step 4 and 5, so we don't need to worry about them yet. The tapped bits denote the indices of the register which will be sampled once the register is clocked. Their values are sent through the XOR operator and fed back to the register. Figure 4 visualizes the 3 registers, the positions of tap-bits, clock-bits and their linear feedback functions.

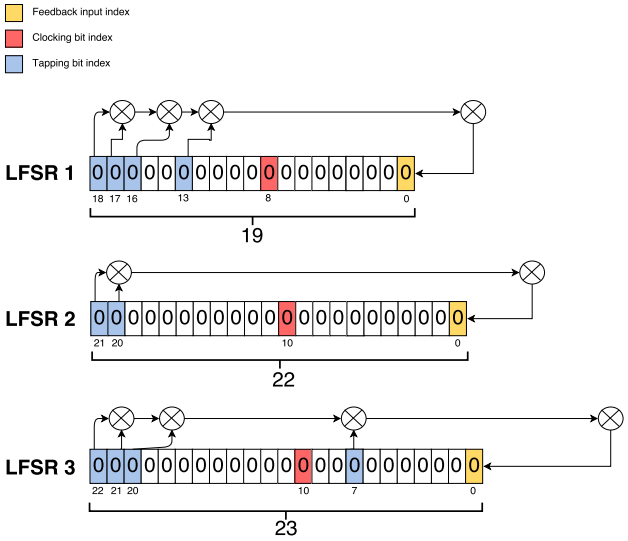

Figure 4: Initial LFSRs

3.2.2 Step 2 - Clocking LFSRs with session key

The registers are initialized with zero-values, and are in this step clocked 64 times, one time for each bit in the session key. The bits of the 64 bit session key are consecutively XORed in parallel with the feedback of the register, and the result is fed into the LSB of the respective register. Figure 5 shows the LFSR configurations and the session key. Note that the figure displays the initial state of step 2. The first value of the session key is highlighted in red and is ready to be fed into the circuit. Each cycle will result in a new, unique state, and this step of the A5/1 algorithm is finished when the last bit of the session key is fed into the circuit.

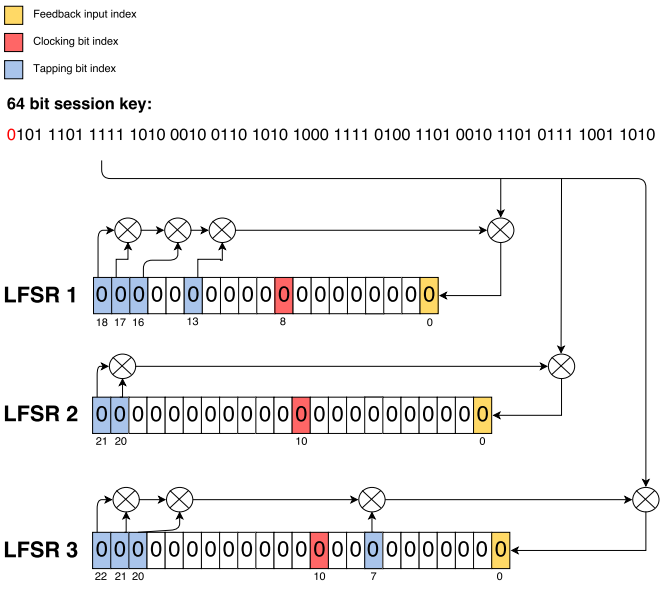

Figure 5: Clocking LFSRs with session key

3.2.3 Step 3 - Clocking LFSRs with frame counter

After the registers have been clocked with the session key, they no longer hold only zero-values. Figure 6 shows the resulting states of step 2, and the initial states of step 3. In step 3, the registers are clocked against a 22 bit frame counter, where the bits of the frame counter are XORed with the feedback of each register and fed into the LSB of its respective register. This step is very similar to step 2, where the only difference is the 22 bit frame counter versus the 64 session key. Since we iterate over each bit in the frame counter, this step only clocks the registers 22 times, as opposed to the 64 cycles in step 2.

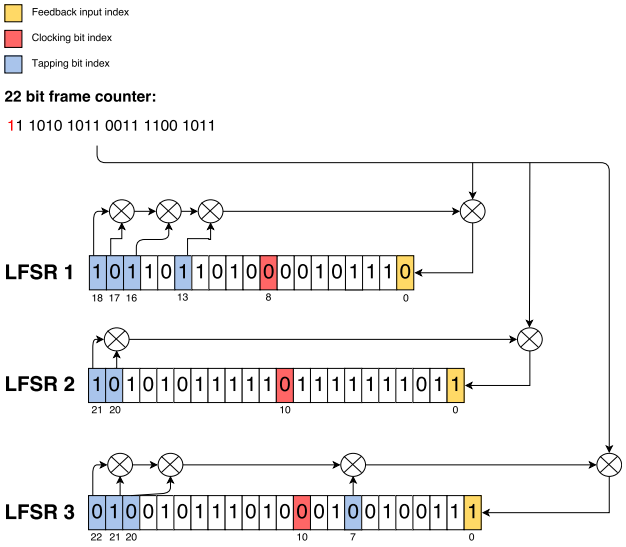

Figure 6: Clocking LFSR with frame counter

3.2.4 Step 4 - Clocking LFSRs with majority vote

We are at this point done with any external input (e.g. session key, frame counter), but the registers still need some more pre-processing. In step 4, the registers are clocked 100 times with irregular clocking. This is where the clock bits come into play. Irregular clocking follows the majority rule, a decision rule that selects alternatives that have a majority. The majority bit is determined by the clocking bits of the registers (LFSR 1 clocking bit: 8, LFSR 2 clocking bit: 10, LFSR 3 clocking bit: 10). If a clocking bit of a register is equal to the majority bit, the register is clocked. Otherwise, the register is left unchanged.

Figure 7 and figure 8 show two cases of the majority rule, and how it affects the clocking decision. Note, both cases are arbitrarily chosen to easily explain the majority rule and are not derived from the algorithm. In figure 7 the majority bit is decided by the values (0,1,0) which are the clocking bits of the respective registers (marked in red). $Maj(0, 1, 0) = 0$ is the majority bit of the LFSRs clock bits. The LFSRs whose clock bit is equal to the majority bit are clocked, while the last LFSR is left

unaltered.

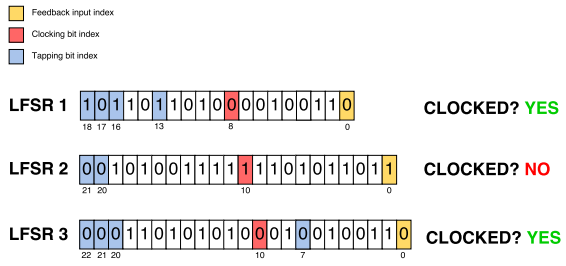

Figure 7: Case 1: Irregular clocking of LFSR 1 and LFSR 3

Figure 8 visualizes a scenario where the majority bit is $Maj(0, 1, 1) = 1$. Thus LSFR 2 and LSFR 3 are clocked, while LSFR 1 is left alone.

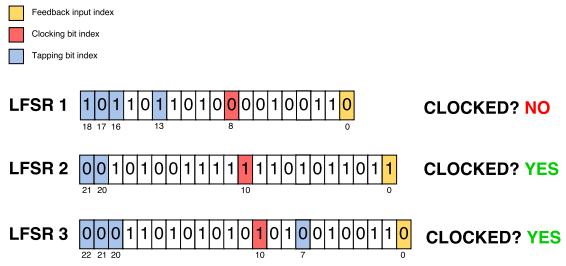

Figure 8: Case 2: Irregular clocking of LFSR 2 and LFSR 3

More general, we can showcase the majority rule and its effect on the clocking of registers with the figure below (figure 9), where **CB** is an abbreviation for 'clocking bit'.

CB of $R1$**:** $R1[8]$	0	0	0	0	1	1	1	1
CB of $R2$**:** $R2[10]$	0	0	1	1	0	0	1	1
CB of $R3$**:** $R3[10]$	0	1	0	1	0	1	0	1
R1 clocked?	√	√	√	–	–	√	√	√
R2 clocked?	√	√	–	√	√	–	√	√
R3 clocked?	√	–	√	√	√	√	–	√

Figure 9: Clockcontroll of A5/1

The initial state of step 4 is shown in figure 10. As mentioned, the entire system will be clocked 100 times with irregular clocking, which will alter the states of the registers. The resulting state will be displayed in figure 11.

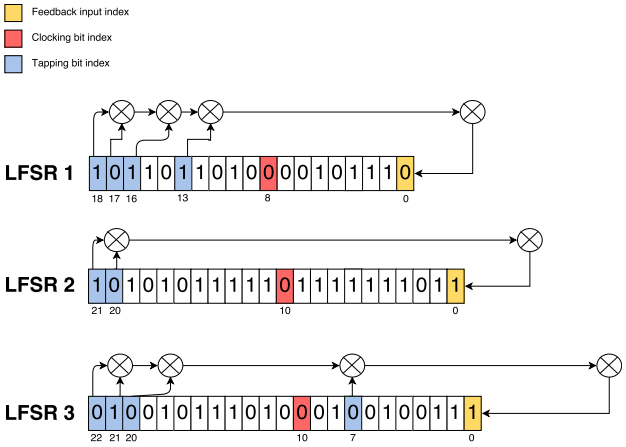

Figure 10: Bank currency exchange

3.2.5 Step 5 - Production of key stream

Pre-processing of the registers is now complete, and they are ready to produce the pseudo random bit stream that will encrypt the plain text. To produce the PRAND bit stream, called "key stream", the registers are clocked 228 times with irregular clocking. For each cycle, the most significant bit (MSB) of each register is used for a final computation. Each register's MSB is XORed with one-another, and the result is added to the key stream. Figure 11 shows the additional XOR logic gate that calculates the key bit. Since each cycle generates a key stream bit, and we cycle our circuit 228 times, the output key stream will consist of 228 bits.

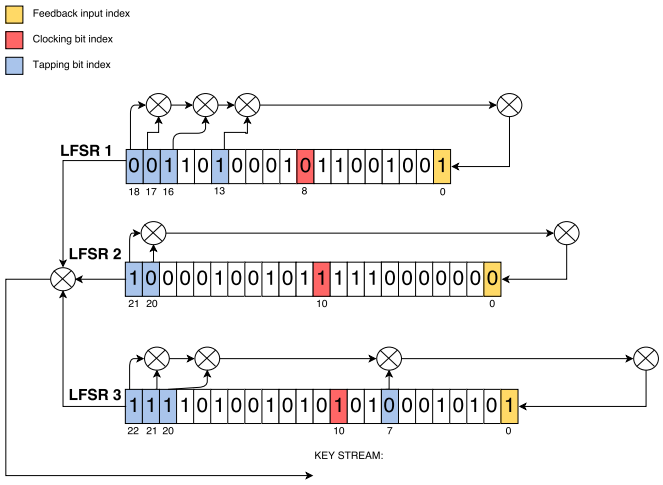

Figure 11: PRAND generation

3.2.6 Step 6 - Creating cipher text

Now that we have a pseudo random generated key stream, we can encrypt our plain text. Recall that only 114 bits of the key stream can be used to encrypt our data, as 114 bits will be used to decrypt the incoming message. To encrypt our data, we take a 114 bit chunck of plain text and XOR it bit for bit with the 114 bits of key stream dedicated for encryption. This simple XOR

between plain text and key stream creates the cipher text, which we can send securely over the GSM network. Since the key stream is pseudo random, the cipher text will appear random as well.

4 Vulnerabilities in A5/1

4.0.1 Known attacks and vulnerabilities

Pioneering contributions to the analysis of A5/1 have been provided in several papers. The known attacks can be summarized in the following way:

- Briceno[4] discovered that in all the deployed versions of the A5/1 algorithm, the 10 least significant bits of the 64 bit key were always set to zero, which reduced the key-search, and the problem drastically.
- Anderson and Roe[5] proposed an attack based on guessing the 41 bits in the shorter R1 and R2 registers, and deriving the 23 bits of the R3 register from the output. If we assume that a PC can test ten million guesses per second, Anderson and Roe's would still need more than a month to find one key.
- GoliC[3] designed a complex attack, where each step is based on the solution of a system of linear equations. Even though the attack requires fewer steps, each step is more complex, and does not run run faster than previous attacks on a regular computer.

The listed attacks require between $2\hat{4}0$ and $2\hat{4}5$ steps, to break the A5/1 algorithm. This level of security makes the algorithm vulnerable to hardware-based attack, but not software-based. Alex Birkyukov, Adi Shamir and David Wagner proposed a real time cryptanalysis of A5/1 on a PC[8]. In their paper, they describe their attack which is based on flaws in the tap structure[3], they showed that A5/1 can be decrypted in real-time. Most of the attacks exploit the fact that the size of the key is small and that it can be shortened down enough to be brute forced. Flaws in the design of GSM Authentication has also led to other types of attacks, often through IMSI-Catchers.

4.0.2 IMSI-Catchers

An IMSI-Catcher is a virtual base transceiver station[4], a device to identify the IMSI[5] of a cellular device connecting to the GSM network[7]. The IMSI-catcher acts as a base station and can thus log all the IMSI-Numbers of all devices nearby. By emitting a strong signal, the IMSI-Catcher can make sure to be the preferred base station and can therefore perform man-in-the-middle attacks. The simple GSM specifications only require one-way authentication, namely the network must authenticate the cellular device. Thus, since the base station is never authenticated, the cellular device is very vulnerable towards attacks. The authentication issue allows the IMSI-Catcher to downgrade the preferred encryption scheme to a less secure one. Even downgrades to the unencrypted A5/0 algorithm can be achieved. This type of attack allows an adversary to easily decrypt cipher texts, as the plain text is encrypted with an insecure scheme. Therefore, succeeding technologies like UMTS(3G) and 4G requires two-way authentication and can not be exploited as easy.

5 Conclusion

The GSM network was the first mobile cellular network with an aim to connect the world. The new requirements for over-air security were not fulfilled by the original 2G implementations. Short encryption keys, flaws in the A5 tap-structure, as well as one-way authentication are some of the vulnerabilities that GSM is faced with. Technologies based on GSM have fixed many of these issues, but there are still ways to downgrade the security preferences in many cases.

References

[1] Internet Archive. Gsm: Global system for mobile communication. `https://web.archive.org/web/20140208025938/http://www.4gamericas.org/index.cfm?fuseaction=page&sectionid=242`. [Online; accessed 12-June-2017].

[2] Unknown Author Wikipedia Article. Security through obscurity. `https://en.wikipedia.org/wiki/Security_through_obscurity`. [Online; accessed 12-June-2017].

[3] J. Golic. Cryptanalysis of alleged a5 stream cipher. *proceedings of EUROCRYPT'97, LNCS 1233,pp.239255, Springer-Verlag 1997*, May 1999.

[4] D. Wagner M. Briceno, I. Goldberg. A pedagogical implementation of a5/1. May 1999.

[5] M. Roe R. Anderson. A5. *http://jya.com/crack-a5.htm*, 1994.

[6] SANS Institute InfoSec Reading Room. The gsm standard (an overview of its security). `https://www.sans.org/reading-room/whitepapers/telephone/`

[4] VBTS - Virtual Base Transceiver Station is a virtual abstraction of the physical device that facilitates the wireless connection with the cellular device and connects it to the cellular network

[5] IMSI - International Mobile Subscriber Identity

gsm-standard-an-overview-security-317. [Online; accessed 12-June-2017].

[7] Unknown. Imsi-catcher. https://en.wikipedia.org/wiki/IMSI-catcher. [Online; accessed 12-June-2017].

[8] Adi Shamir Alex Biryukov David Wagner. Real time cryptanalysis on a5/1 on a pc. https://cryptome.org/a51-bsw.htm. [Online; accessed 12-June-2017].