Security Enhancements for A5/1 Without Loosing Hardware Efficiency in Future Mobile Systems

[*]N. Komninos, [*]B. Honary, [**]M. Darnell

[*]Lancaster University (UK)
[**]HW Communications Limited (UK)

Abstract

A5/1 is the strong encryption algorithm which protects the air interface of the GSM cellular network. However, in the Fast Software Encryption Workshop 2000 two attacks, the biased birthday attack and the random subgraph attack against A5/1 were presented [1]. In this paper, we propose new security enhancements to improve A5/1 encryption algorithm from the biased birthday attack [1] and random subgraph attack [3] [4] in order to be used in future mobile communication systems. The improvements that make both attacks impractical are based on the clocking mechanism of the registers, and their key setup routine. Furthermore, we have increased the linear complexity of A5/1 to make the solution of the linear equations [2] impractical in real time systems. Finally, both original and modified versions of A5/1 were implemented easily in a CPLD device for 3rd generation mobile systems.

Introduction

Encryption in future mobile systems is essential to protect sensitive information and prevent fraud. Furthermore, future mobiles phones require an efficient encryption algorithm which is secure, small in size, fast and easy to implement in hardware. Nowadays, the most efficient stream cipher is A5/1, which protects the air interface of the GSM cellular network. A5/1 is fast, small in size and suitable to be implemented in future mobile terminals. However, in the Fast Software Encryption Workshop 2000, the biased birthday attack and random subgraph attack against A5/1 were presented [1]. They are based on subtle flaws in the tap sequence of the registers, their noninvertible clocking mechanism, and their frequent resets. When these attacks are applied, cryptanalysis of A5/1 is performed on a single PC.

Description of A5/1

Each conversation in GSM is sent encrypted by a new session key K as a sequence of frames every 4.6 millisecond. For each frame, K is mixed with a frame counter F_n and the result serves as the initial state of A5/1 which produces 228 pseudo random bits. These 228 bits are used to encrypt/decrypt traffic between the mobile station and the base station.

According to [1] and [2] A5/1 is built from three linear feedback shift registers (LFSR) of lengths 19, 22, and 23 bits, which are denoted by R1, R2, and R3 respectively. The taps of R1 are at bit positions 13, 16, 17, 18; the taps of R2 are at bit positions 20, 21; the taps of R3 are at bit positions 7, 20, 21, 22. The three LFSR's with number of states $n = 2^{64}$ are maximum length with period $2^{19}-1$, $2^{22}-1$, and $2^{23}-1$. Each register has a single clocking tap in bit 8 for R1, bit 10 for R2, and bit 10 for R3; each clock cycle, the registers whose clocking taps agree with the majority bit are actually clocked

(Figure 1). Note, that at each clock cycle two or three registers are shifted and that each register moves with probability $3/4$ and stops with probability $1/4$.

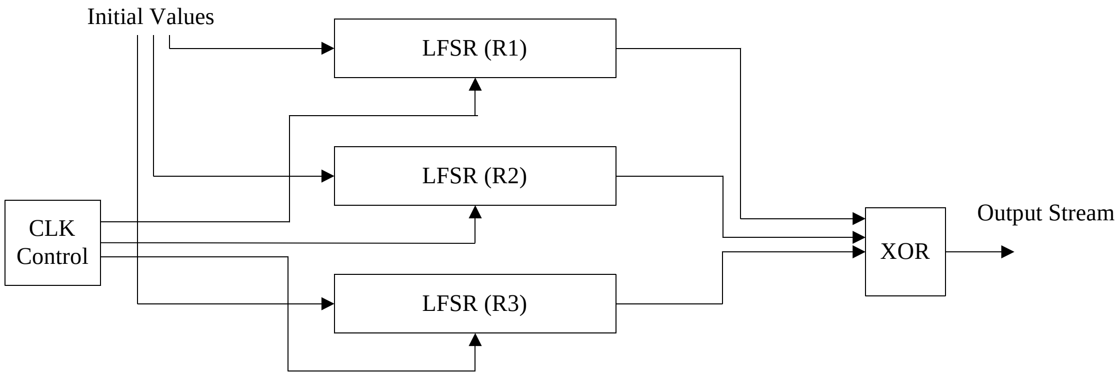


Figure 1 – A5/1 Encryption Algorithm

The initial state of the generator is carried out in the following way: R1, R2, and R3 are first zeroed and then clocked for 86 cycles ignoring the stop/go clock control. During this period each bit of K is XOR'ed in parallel into the lsbs of the three registers for the first 64 clock cycles. Then F_n is XOR'ed in parallel into the lsbs of the three registers for the remaining 22 clock cycles. Finally, the three registers are clocked with the stop and go control for 328 additional cycles. No output is produced in the first 100 clock cycles. In the next 228 clock cycles 228 bits are produced by XOR'ing the three msbs of R1, R2, and R3.

SOLUTIONS TO THE ATTACKS

The main idea of the biased birthday attack [1] is to consider sets A and B which are not chosen with uniform probability distribution among all the possible states. The observation which makes this attack efficient is that in A5/1 there is a huge variance in the weights of various states that begin with a specific 16-bit pattern of bits named a. The register bits that affect clock control and the register bits that affect the output are unrelated for about 16 clock cycles. This decreases the states to be sampled to $2^{64} * 2^{-16} = 2^{48}$. It was also found that the weight of about 85% of the states was zero because their trees died out before reaching depth 100. Other weights ranged from 1 to more than 26,000 [1]. As a result, efficient determination of initial states can be obtained since the exact location of a and the depth of the initial state is known. This is made possible by the *initial mixing of the key and frame number*, which ignore the usual clock control, and the *poor choice of the clocking taps*.

Assume that we know the state of A5/1 immediately after the key and frame counter were used, and before the 100 mixing steps. It was found in [1] that by running backwards, the frame counter can be eliminated in a unique way, and 64 linear combinations of the 64 key bits can be obtained. The correct key can be chosen when running A5/1 forward for more than 100 steps, and comparing the results with the actual data in the next frame.

In [1] it was also found that the majority clock control rule implies that up to 4 states can converge to a common state in one clock cycle, and some states have no predecessors. Furthermore, the average number of predecessors of each node is 1, and thus the expected number of vertices in the first *k* levels of each tree grows only linearly in *k* [3].

New clock control mechanism[1] and key setup routine are proposed to overcome the above problems and to increase the complexity of the algorithm without loosing hardware efficiency. In the new clock control mechanism each register has two clocking taps in bits 5, 11 for R1, bits 6, 13 for R2, and bits 7, 14 for R3. The clocking taps divide each register in three almost equal parts. Every clocking tap per register is checked in a triangle scheme; bits 5, 13, and 7 of R1, R2, and R3 respectively; bits 11, 6, and 14 of R1, R2, and R3 respectively as shown in Figure 2.

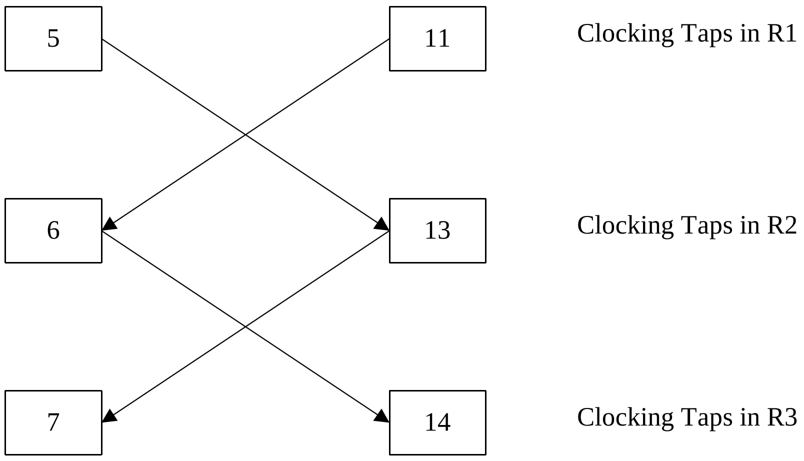


Figure 2 – Clock Control Mechanism

Each clock cycle, the registers whose clocking taps agree with the majority bit in both triangles are shifted. For example, if A, B, and C represent the clocking taps of R1, R2, and R3 respectively then table 1 shows all the combinations for shifting.

One Clock Cycle	**Majority Bit of 5,13,7=A,B,C**	**Majority Bit of 11,6,14=A,B,C**	**Shift**
Case 1	A=B	A=B	R1, R2
Case 2	A=C	A=B	R1
Case 3	B=C	A=B	R2
Case 4	A=B	A=C	R1
Case 5	A=C	A=C	R1, R3
Case 6	B=C	A=C	R3
Case 7	A=B	B=C	R2
Case 8	A=C	B=C	R3
Case 9	B=C	B=C	R2, R3
Case 10	A=B	A=B=C	R1, R2, If 7=14 then R3
Case 11	A=C	A=B=C	R1, R3, If 13=6 then R2
Case 12	B=C	A=B=C	R2, R3, If 5=11 then R1
Case 13	A=B=C	A=B=C	R1, R2, R3

Table 1 – Shifts per Clock Cycle

Note, that at each clock cycle one or/and two or/and three registers are shifted and that each register moves with probability $\cong 8/13$. In this case the register bits that affect the clock control and the register bits that affect the output are unrelated for only 6 clock

[1] Patent Pending; Patent Application Number 0123302.2

cycles instead of 16 in the clock control of A5/1. This increases the number of states required to be sampled in A5/1 from $2^{64} * 2^{-16} = 2^{48}$ to $2^{64} * 2^{-6} = 2^{58}$.

Moreover, in the new key setup routine, the key and the frame counter are mixed in the following way: the three registers are zeroed and clocked for 88 cycles including the clock control mechanism described earlier. During this period each bit of K and F_n is XOR'ed (from lsb to msb) in parallel with the lsb of the three registers. Finally, the two results are XOR'ed again to determine which of K or F_n will be inserted in the lsb of the three registers. If the final XOR is '1' then K will be inserted, otherwise, F_n. After the initial state, the key setup routine can follow the original procedure. In the proposed routine, the session key is mixed with the frame counter in a random way which makes it difficult to be extracted from the initial state.

Likewise, the main idea of the random subgraph attack is to make most of the special states accessible by simple computations from the subset of special states which are actually stored in the hard disk [1]. The attack is based on a function f that maps special states in an easy computable way and Hellman's time-memory tradeoff, $M\sqrt{T} = |U|$, for block ciphers described in [4] can be used to invert it easily. In A5/1 the attack is applied to the subspace of 2^{48} by the fact that it can be efficiently sampled. The time trade-off formula of the random subgraph attack results to $M = 2^{36}$ and just $T = 2^{24}$. This number of steps can be carried out in several minutes on a fast PC.

The clocking control mechanism proposed above makes this attack inefficient because states cannot be sampled efficiently any more. The memory and time required are $\left(M = 2^{36}\right)$ and $\left(T = 2^{40}\right)$ respectively which is much worse than previous attacks [2] [3].

Furthermore, the proposed key setup routine was carried out and it was found that only 1% of the states were zero because their trees died out before reaching depth 100. This is happening because of the frequent initialisations. In this case the biased birthday attack becomes very inefficient which results to an impractical random subgraph attack.

Finally, we have also increased the linear complexity of A5/1, by adding an AND gate to R2 and R3, from 128 to 570 in order to prevent practical attacks [1] to be applied efficiently. Actually, different combinations of boolean functions can be included in the three registers to make A5/1 mathematically more complex [2]. In every clock cycle R2 and R3 are AND'ed and the result is XOR'ed with R1, or/and R2, or/and R3. Increasing linear complexity in a cipher with small number of states prevents attacks described in [2] as well as in [1].

The linear complexity of A5/1 is calculated in the following way:
Let x_1, x_2, and x_3 denote the element of the polynomial with highest degree in R1, R2, and R3 respectively. The output y of A5/1 is given by the following equation:

$$y = x_1 \oplus x_2 \oplus x_3 \quad (1)$$

If L_1, L_2, and L_3 denote the highest degree of the primitive polynomials in R1, R2, and R3 respectively, then according to (1) the linear complexity of A5/1 is given by [6]:

$$L = L_1 + L_2 + L_3 = 19 + 22 + 23 = 128$$

However, the linear complexity is larger in the improved version because of the AND gate. Thus:

$$y = x_1 \oplus x_2 \oplus x_3 \oplus x_2 \cdot x_3 \quad (2)$$

According to (2)

$$L = L_1 + L_2 + L_3 + L_2 L_3 = 19 + 22 + 23 + 506 = 570$$

Of course, the linear complexity is even greater than ≥ 570 because of the new control mechanism used (Figure 2).

Implementation / Results

Both versions of A5/1 were implemented on a complex programmable logic device (CPLD) using VHSIC Hardware Description Language (VHDL). The XC9572 Xilinx chip was used to implement the A5/1 encryption algorithm. XC9572 has 72 macrocells and registers (about 1,600 gates) out of which 65 (90%) and 64 (88%) were used for both versions respectively. Thus, in our implementations the A5/1 occupied about 1,444 gates in XC9572. The internal structure of XC9572 in Figure 3 shows the resources that were used. The minimum clock cycle period and the maximum speed in the original A5/1 in XC95216 were 31.0ns and 32.2MHz respectively.

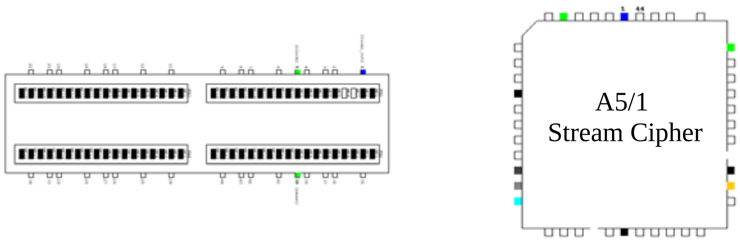
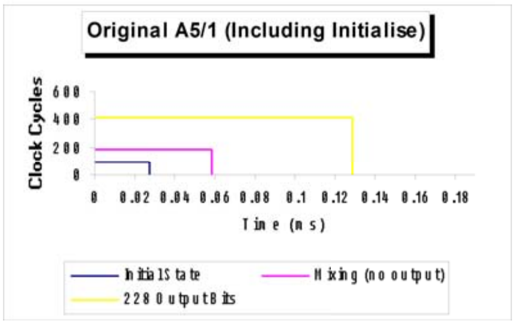


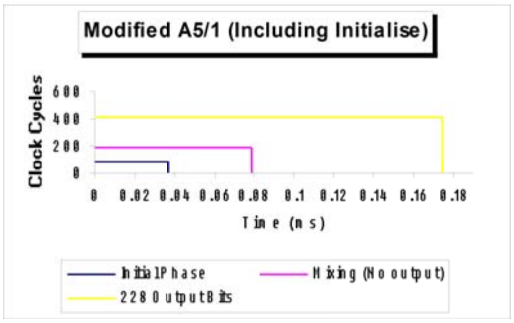
Figure 3 – Internal Structure of XC9572 chip for A5/1

In the improved version of A5/1, the size of the chip was the same (Figure 4) since only one AND gate was added to the original design (Figure 1). However, the minimum clock cycle period and the maximum speed of the modified version were 42.0ns and 23.8MHz respectively. The delay in the minimum clock period and the maximum speed of the chip was also expected once the digital logic was increased with the key setup routine and the new clock control mechanism. The total delay, as shown in Figure 4c, is very small and it does not exceed GSM specifications where 228 bits must be generated within a time slot duration (i.e. 0.6ms).

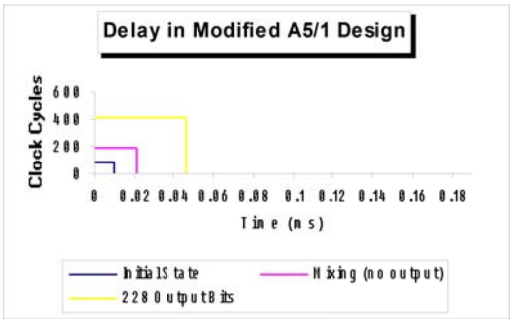
In 3rd generation mobile systems (UMTS) the length of the frame is 10ms and each time slot occupies 0.625ms. Even though we increased the number of LFSRs in A5/1, which resulted to an increase of the key size from 64 to 128, and applied the same key setup routine and clock control the results obtained were very promising. Based on the results the new stream cipher is fast enough to be implemented in the UMTS mobile terminals.



(a)



(b)



(c)

Figure 4 – (a) Original A5/1 (b) Modified A5/1 (c) Total Delay in Modified A5/1

In the original A5/1 implementation, the initial, mixing, and output states require 0.02728ms, 0.05828ms, and 0.12896ms respectively (Figure 4a). On the other hand, in the modified implementation the initial, mixing, and output states need 0.03696ms, 0.07896ms, and 0.17472ms respectively (Figure 4b). As shown in Figure 4c, the delay between the two implementations is 0.00968ms, 0.02068ms, and 0.04576ms for the initial, mixing, and output states respectively. This delay is not so critical because in both GSM and UMTS specifications the limit is about 0.6ms.

CONCLUSION

A5/1 has been designed to be fast, easy to implement, and most important small in size according to the GSM specifications. However, it has proved that the encryption algorithm provided by the GSM is not very secure [1] [2]. The security enhancements we have proposed in the new clock control mechanism increase the level of security from 2^{40} to 2^{56} steps based on current attacks and in the proposed key setup routine the session key can be not extracted easily. In this paper, we have improved the level of security in A5/1 without compromising size. Even though the improved version is slower the delay is not significant and most important does not exceed UMTS specifications.

The improved version of A5/1 prevents special states to be identified by their output sequences [1]. The clock control mechanism described prevents that since the register bits that affect the clock control and the register bits that affect the output are unrelated for only 6 clock cycles. This makes the biased birthday attack and the random subgraph attack [1] inefficient. Moreover, Anderson and Roe [5] have proposed an attack, in A5/1 that uses only 3 LFSR, based on guessing the bits of the shorter registers, and deriving the other bits of the longer register from the output. However, they have to guess the clocking sequence and the complexity of the attack is greater than $\geq 2^{56}$.

Finally, in the proposed key setup routine, the session key is mixed with the frame counter in a random way which makes it difficult to be extracted from the initial state.

ACKNOWLEDGEMENTS

Special thanks to HW Communications Limited for their financial support.

REFERENCES

[1] A. Biryukov, A. Shamir, D. Wagner, *Real Time Cryptanalysis of A5/1 on a PC*, Fast Software Encryption Workshop 2000, New York, USA, April 10-12, 2000.

[2] J. Golic, *Cryptanalysis of Alleged A5 Stream Cipher*, proceedings of EUROCRYPT'97, LNCS 1233, pp.239-255, Springer – Verlag 1997.

[3] S. Babbage, *A Space / Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers*, European Convention on Security and Detection, IEE Conference publication, No 408, May, 1995.

[4] M. E. Hellman, *A Cryptanalytic Time – Memory trade-Off*, IEEE Transactions on Information Theory, Vol. IT – 26, N 4, pp.401-406, July 1980.

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

[6] A. J. Menezes, P.C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press LLC, 1997.