Cryptanalysis of the A5/1 GSM Stream Cipher *
NES/DOC/TEC/WP3/005/a

Eli Biham † Orr Dunkelman ‡

Abstract

A5/1 is the stream cipher used in most European countries in order to ensure privacy of conversations in GSM mobile phones. In this paper we describe attacks on this cipher with total work complexity $2^{39.91}$ of A5/1 clockings, given $2^{20.8}$ known plaintext. This is the best known result with respect to the total work complexity.

1 Introduction

The GSM mobile phones system, which has more than 200 million users, uses a built-in encryption to protect the privacy of conversations. Two ciphers are used. A5/1 is used in most European countries, and A5/2 is used in most other countries. Both ciphers, combine a few (3 or 4) Linear Feedback Shift Registers (LFSR), whose clocks are controlled by some of their bits.

A5/1 and A5/2 were kept secret by the GSM companies for a long period of time. Only recently A5/1 and A5/2 were reverse-engineered and published by Brinceno, Goldberg and Wagner at [4, 5]. Afterwards A5/2 was also cryptanalyzed by Wagner [5].

Before A5/1 was reverse-engineered, several researchers and in particular Ross Anderson, identified and published [1] the general structure of A5/1. Jovan Golic attacked in [7] this structure with parameters which are very close to the actual parameters of A5/1 by solving a system of 2^{41} linear equations. This attack is likely working on the real version of A5/1. The complexity of both attacks is equivalent to about 2^{47} A5/1 clockings.

A5/1 is a stream cipher with 64-bit key. Encryption is also controlled by a 22-bit frame number which is publicly known. In practice, A5/1 is always used

*This work was supported by the European Union fund IST-1999-12324 - NESSIE and by the Technion's Chais' Excellence Program.

†Computer Science department, Technion - Israel Institute of Technology, Haifa 32000, Israel, biham@cs.technion.ac.il, http://www.cs.technion.ac.il/~biham/.

‡Computer Science department, Technion - Israel Institute of Technology, Haifa 32000, Israel, orrd@cs.technion.ac.il, http://vipe.technion.ac.il/~orrd/me/.

Register Number	Length in bits	Primitive Polynomial	Clock-controling bit (LSB is 0)	Bits that are XORed
1	19	$x^{19}+x^5+x^2+x+1$	8	18,17,16,13
2	22	$x^{22}+x+1$	10	20,21
3	23	$x^{23}+x^{15}+x^2+x+1$	10	22,21,20,7

Table 1: The A5/1 Registers Parameters

with only 54 bits of key, with the remaining 10 bits set to zero. This indicate that our results (which are based on 64-bit key) might even be improved for the current applications of the cipher.

Biryukov and Shamir had presented an improvement to Golic Time-Memory tradeoff attack in [2]. The attack requires about 2 minutes of GSM conversation, and finds the key in few seconds. However, their attack requires a pre-processing phase equivalent to 2^{42} A5/1 clockings and four 73 GB hard-drives, or 2^{48} A5/1 clockings with two 73 GB hard-drives.

In this paper, we introduce a new technique for retrieving the internal state of the cipher. Given the internal state of the cipher it is easy to find the secret key by clocking the cipher backwards, using the technique presented in [2].

This paper is organized as follows: In Section 2 we describe the A5/1 algorithm. In Section 3 we shortly describe the previous analysis of this cipher. In Section 4 we describe the basic idea behind our attack. In Section 5 we show improvements to the attack which requires $2^{20.8}$ known output stream, and only $2^{39.91}$ steps of analysis (the time unit is one A5/1 clocking). The attack finds the full internal state of the registers, from which the original key can be derived easily, as presented in [2].

2 A Description of A5/1

A5/1 combines 3 LFSRs. Each new step, 2 or 3 LFSRs are clocked, according to a clocking mechanism we describe later. The output is the parity of the outputs of the 3 LFSRs.

We denote the LFSRs as R_1, R_2 and R_3. The lengths of R_1, R_2 and R_3 are 19, 22 and 23 bits, respectively. The output of each LFSR is the last bit (we refer those as bits 18, 21, 22, respectively). The registers are updated according to their primitive polynomials, which are summarized in Table 1. The clocking decision is based upon one bit of each register. The three bits are being extracted (bit 8 from R_1, bit 10 from R_2 and bit 10 from R_3) and their majority is calculated. The two or three registers whose bit agrees with the majority are clocked.

We denote by $R_i[j_1, \ldots, j_l]$ the bits $j_1, \ldots, j_l$ of register R_i.

The initialization of the registers loads the bits of secret key Key, followed by the bits of the frame number $Frame$ and discarding 100 output bits, as follows:

1. Set all LFSRs to 0 ($R_1 = R_2 = R_3 = 0$)
2. For $i := 0$ to 63 do
   (a) $R_1[0] = R_1[0] \oplus Key[i]$
   (b) $R_2[0] = R_2[0] \oplus Key[i]$
   (c) $R_3[0] = R_3[0] \oplus Key[i]$
   (d) Clock all three registers (i.e., for $j > 0$ $R_i[j] \leftarrow R_i[j-i]$, and $R_i[0]$ is set to the result of the primitive polynomial on the previous value of R_i)
3. For $i := 0$ to 21 do
   (a) $R_1[0] = R_1[0] \oplus Frame[i]$
   (b) $R_2[0] = R_2[0] \oplus Frame[i]$
   (c) $R_3[0] = R_3[0] \oplus Frame[i]$
   (d) Clock all three registers
4. For $i := 0$ to 99, clock the cipher by its regular clocking mechanism, and discard the output

After the initialization, 228 bits of output stream are being computed. 114 bits are used to encrypt data from the center to the mobile phone, and the other 114 bits are used to encrypt data from the mobile phone to the center.

We put the figure of the cipher at Figure 1.

3 Previous Work

Several papers about the A5/1 were published [7, 1, 2]. One of them [7] attacks an alleged version, which is very similar to the real A5/1. This attack takes on average $2^{40.16}$ workload and finds the internal state of the cipher. However in [7] the time unit is the time needed to solve a linear equations system, a unit which we do not use. We, on the other hand, use (like [2]) workload unit of A5/1 one clocking. Golic also presents a time-memory tradeoff, which was enhanced by Biryukov and Shamir in [2] after the first version of our paper was written.

Golic's first attack is briefly described as follows: The attack is based on creating 63.32 linear equations (in average, however, the analysis refers to 64 equations) which could be solved and thus retrieving the internal state (as only $2^{63.32}$ internal states are possible).

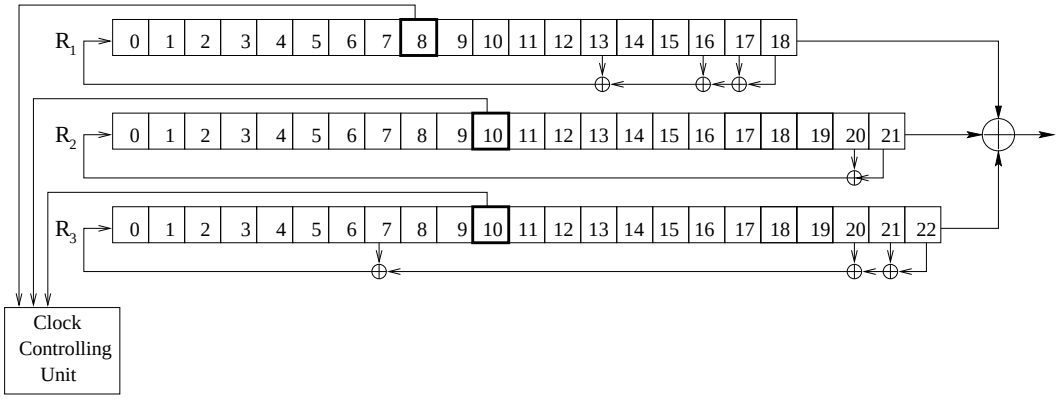

Figure 1: The A5/1 Structure

In the first step, the attacker guesses n bits from all three registers. After this is done, the attacker knows all three registers output, thus on the average he receives $4n/3$ linear equations about the registers contents. Also the first output bit is known to be the parity of all most significant bits from each register in the beginning, thus the attacker obtain another equation. Therefore, the attacker has now $3n+4n/3+1$ linear equations. As in the process of analysis we assumed the bits to be independent, thus, n cannot be bigger than the shortest distance between the clock controlling bit and the output bit.

For $n = 10$ the attacker can get 44.33 linear equations, thus he needs about 19 more equations. In that point Golic noticed that not all 2^{19} options need to be considered. The attacker builds a tree with all the valid options for the possible values for the three input bits to the majority clock-control function. The number of options (for the three bits) is 2.5, as in 3/4 of the cases, two new bits are considered, and in the remaining 1/4 of the cases, 3 new bits are considered. Hence, each node contains $3/4 \cdot 4 + 1/4 \cdot 8 = 5$ options. We now use the knowledge of the output, to discard about half of the options (as they lead to wrong output value). Thus each node has a branching level of 2.5 on the average, and as the knowledge of $4/3m$ bits is sufficient to receive the linear equations about the m bits out of each register (due to the clocking mechanism), the tree of clocking options need to be considered only till the depth of $4/3 \cdot 19/3 = 76/9 = 8.44$. As each level has a branching factor of 2.5, the amount of time needed to search the tree is $2.5^{8.44} \approx 2^{11.16}$.

Hence, the total number of systems is $2^{40.16}$ and solving them should be suffice to retrieve the internal state. Note that the complexity of the attack is $2^{40.16}$ linear equation set solving.

In [3] Biryukov and Shamir presented a time-memory tradeoff attack. In

their attack a special pattern is chosen, and the attacker preprocess many samples which generate this specific long pattern. Once the pattern occurs in the real output stream, the attacker finds the possible internal states, according the rest of the output stream and the pre-processed table. The attack requires 2^{38} to 2^{48} pre-processing time, and has many trade-offs between the length of the conversation (and analysis), the pre-processing phase and the memory needed for the attack.

4 The Basic Attack

The main idea behind the attack is to wait until an event which leaks a large amount of information about the key occurs and then exploit it. Assume that for 10 consecutive rounds the third register (R_3) is not clocked. In such a case, we gain about 31 bits of information about the registers. Assume we know $R_3[10]$ (the clock controlling bit) and $R_3[22]$ (the output bit), then we know for these 10 rounds that all the clock controlling bits in the other two registers are the complement of $R_3[10]$ (thus receiving 20 bits, a bit from each register each round). We also know that the output bit of the stream XOR $R_3[22]$ equals to the output bits of R_1 XOR R_2 (each round). There are at least 11 rounds for which the same output bit is used from R_3, thus we receive another 11 linear equations about the registers. The equations are the parity of the couples $R_1[t] \oplus R_2[t+3]$ for $t = 8, \ldots, 18$.

Guessing 9 bits from R_1 ($R_1[18, 17, 16, 15, 14, 12, 11, 10, 9]$) and another one from R_2 ($R_2[0]$) uncovers all of R_1 and R_2. Note that we do not need to guess $R_1[13]$ as we know the parity bit $R_1[18] \oplus R_1[17] \oplus R_1[16] \oplus R_1[13]$, thus guessing $R_1[18], R_1[17], R_1[16]$ gives $R_1[13]$, in addition to the corresponding bits of R_2 ($R_2[21, 20, 19, 16]$). $R_2[11]$ can be easily computed as $R_1[8] \oplus R_2[11]$ is known (from the output stream), and $R_1[8]$ is known (as it was the first clock-controlling bit in register R_1).

The complexity so far is the cost of guessing twelve bits ($R_3[10]$, $R_3[22]$, $R_2[0]$, and 9 bits of R_1), and we recover all the bits of R_1 and R_2. We can continue to clock the states until R_3 moves, and gain another bit ($R_3[21]$).

By further guessing the bits $R_3[0], \ldots, R_3[9]$ we can continue clocking the third register and receive all the unknown bits of R_3 (total of 11 unknown bits). The workload of this stage is the cost of guessing 10 bits, times the cost of work needed to retrieve the unknown bits for each guess. As each time the register R_3 advances, we recover a new bit of R_3, we need to wait for 12 clockings of R_3 in order to retrieve the full register, and as the probability that the register R_3 advances is 3/4, the expectation is that 16 clocks would suffice to retrieve the whole register. Now we get a guess for the full internal state and need to check for its correctness. The amortized cost for checking whether this is the right state is 2 clock cycles for each guess (all cases need to be clocked at least once, half of the cases should be clocked another time, 1/4 a third time, etc.).

Thus, this stage requires $2^{10} \cdot 2^4 \cdot 2 = 2^{15}$ workload.

The total expected running time of this attack is $2^{12} \cdot 2^{15} = 2^{27}$, given the location where R_3 is static. This location is of course unknown, thus, we need to examine about 2^{20} possible starting locations (from many possible frames). Hence, the basic attack requires about 2^{47} running time and about 2^{20} bits of the output stream.

Note that this complexity is similar to the complexity of earlier attacks (solving 2^{41} linear equations sets in [7] can be done in around 2^{47} time), and represent the basic flaws in the cipher's design, who provide security of only 47 key bits (instead of 64).

5 Trick or Treat?

We now present several techniques to lower the time complexity required for the attack.

The first technique we use to reduce the complexity of the basic attack is based on a technique presented in [3]. It is known that the polynomials of the registers of A5/1 are primitive, and thus the states (except for the all-zero state) form one huge cycle. Selecting any non-zero state and computing the next states iteratively would result with all (non-zero) states. Therefore, for each register we choose a non-zero state, and clock the register until we get the same state again. We store the states in a table in the order they are computed. We call this table the *next-state* table. In that table if entry i has state x, then the consecutive state is stored in entry $i+1$. During the generation of this table, we also generate a table of pointers, in which for each state x we keep its location in the next-state table.

Given the two tables (for each register), we can clock any state s by any number of clockings c in a fixed time. The method for doing so is to access entry s in the pointers table to find the location l of s in the next-state table, and then access entry $l + c$ in the next-state table. The cost of the first access is two original A5/1 clockings (as we access two tables), but once the location in the next-state table is known, only one table needs to be accessed. Thus, when a state should be clocked again (or iteratively) the cost is the same as one original A5/1 clocking.

The memory size needed for the tables is about 71.5 MB. For each state we need to store the next state and pointer to the next-state table. As each of these fields is in the size of the register, the total size of an entry is bounded by $2 \cdot 23 = 46$ bits (or 6 bytes), and there are 2^{23} states in R_3, 2^{22} in R_2 and 2^{19} in R_1. Thus the tables' size is about $(2^{23} + 2^{22} + 2^{19}) \cdot 2 \cdot 23 = 2^{29.16} \approx 71.5 \cdot 2^{20}$ bits which are 71.5 MB.

For any possible value of the 5 most significant bits in each register and the next 5 clock-controlling bits, we calculate as many bits of the future output stream as possible. We build another table indexed by the first 5 bits of the

output stream, and the 20 bits of R_1 and R_2 (5 MSB and 5 clock-controlling from each), which contains in each entry the possible values of the 10 bits from R_3 which generate this 5-bit output stream. Note that for some fraction of the cases more than 5 bits of output stream can be generated (given the 10 bits of each register), thus given the output stream we can reduce the number of possible values of the bits of R_3 which generate the output stream. In order to do so we separate the cases according to the length of the output stream. The first would be all the values of R_3's bits which generate 5-bit output stream, the second would be all the values of R_3's bits which generate 6-bit output stream (separated according to the 6th bit value), and the same for 7-bit output stream (8-bit output stream cannot be gained due to lack of clock controlling bits, as each clocking requires at least two new clock controlling bits, and we have all in all 15 clock-controlling bits). Each entry also contains how many clocks each register needs to be clocked after the given known output stream has been generated by the 30 bits (10 from each register as above) of the state. The computation of this table requires about $2^{30} \cdot 6 \approx 2^{32.6}$ A5/1 clockings, which can be computed in advance.

Given the above tables, the cost of the last stage of the basic attack (recovering R_3) is much lower, as after the 20 bits from R_1 and R_2 are known, given the 5 bits of the output, the table contains on average 2^5 options for the 10 bits of R_3. However, in some cases (where 6-bit or 7-bit output stream can be computed) the table contains less valid options, as we know that if the 6th bit, and some R_3 value suggest the complement, then this suggestion is not true. On average there are $2^{4.53}$ valid options given the output stream. This follows from the fact that the length of output stream is determined by the clock-controlling bits. Looking at the possible 15 clock-controlling bits, 15968 options generate 5-bit output stream section, 14280 options generate 6-bit output stream section, and the rest 2520 generate 7-bit output bit stream section. Therefore, on average given the 20 bits of R_1 and R_2 and the output stream we have $2^{4.53} \approx 23.2$ options for the 10 bits of R_3.

Note that in the first time we access the table 3 bits are known ($R_3[22], R_3[21]$, $R_3[10]$), so we have about $2^{1.53} \approx 2.9$ options for the rest of the $10 - 3 = 7$ bits. As this need be taken into account, we can sort the values in the table according to those 3 bits, and then in one table access retrieve the valid $2^{1.53}$ values directly. Doing that either increases the time of analysis or table size by a small factor.

For each remaining case we clock the registers as many times as needed according to the first table access (as the table contains the number of clockings for each register). Then, we approach the table again and get about $2^{4.53}$ options for 10 unknown bits of R_3.

We now lack 3 bits from R_3. Approaching the original table again would cost us too much (we would get $2^{4.53}$ options and need to check them all), thus we use a smaller table containing only 6 bits of each register. This table should suffice to give us the last 3 unknown bits. This table is expected to give us for

each case 0.88 possible values on average.

In this stage, we discard the wrong options in the regular method of clocking forward (though we can use the table to do this stage more efficiently, however the improvement of the complexity is negilible).

The total time complexity is calculated as follow: There are 2^{20} possible starting places, each has 2^{12} possible guesses for the first stage bits (9 from R_1, $R_2[0]$ and $R_3[10,22]$). For each possibility we get about $2^{1.53}$ possible values for some of the unknown bits of R_3, and the clocking costs 2 clocking units (as this is the first approach to the next-state table). Then, for each possible value for the first 7 unknown bits from R_3, we get about $2^{4.53}$ possible values for the next 10 unknown bits of R_3. Each possibility needed to be clocked once, then accessing the smaller table, and on case that there are possibilities in that table (on average there are 0.88 possible values for the remaining 3 unknown bits), we need to clock twice to check the guess. Thus, the time complexity of the attack is equivalent to $2^{20} \cdot 2^{12} \cdot 2^{1.53} \cdot 2 \cdot 2^{4.53} \cdot (1+1+2 \cdot 0.88) = 2^{40.97}$ A5/1 clockings.

We can improve this result using more bits in the table. If we can build the table based on 12 bits from R_1 and R_2 (6 most significant bits, and 6 clock-controlling bits), and 10 bits from R_3. In that case 23328 options have 5-bit output stream, 59808 6-bit output stream, 41496 7-bit output stream and 6440 8-bit output stream. This way given the 24 bits of register R_1 and R_2 and the 5 bits of the output-stream, there would be only 2^4 options on average for the ten bits of R_3. Thus, the complexity in this case is $2^{20} \cdot 2^{12} \cdot 2^1 \cdot 2 \cdot 2^4 \cdot (1+1+2 \cdot 0.88) = 2^{39.91}$. The preprocessing time is about $2^{34} \times 8 = 2^{37}$ A5/1 clockings, and the table size is $2^{34} \cdot 2 = 2^{35}$ bytes, i.e., 32 GB of data, times the overhead which is of factor 2, i.e., 64 GB.

6 Summary

We have shown a technique to cryptanalyze the A5/1 cipher. The attack is feasible, in the current technology, and it points out that this scheme should be replaced. The attack requires $2^{39.91}$ A5/1 clockings. The attack requires $2^{20} \cdot 228/(228-63) = 2^{20.8}$ bits of data, which are equivalent to about 2.36 minutes of conversation.

The retrieval of the key given the internal state can be easily done using the algorithm presented in [7].

We summarize the our results and the previously known results in Table 2.

References

[1] Ross Anderson, *On Fibonacci Keystream Generators*, Proceedings of Fast Software Encryption - FSE '95, Springer vol. 1008, pp. 346-352, 1995.

Attack	Precomputation Workload	Complexity of Analysis	Time Unit	Data Complexity (bits)	Memory Requirements
[7] - Basic Attack	0	$2^{40.16}$	Linear eq. set solving	64	0
[7] - TM Tradeoff	$2^{35.65}$	$2^{27.67}$	Linear eq. set solving	$2^{28.8}$	862GB
[3] - Baised Birthday Attack	2^{48}	1 second	A5/1 Clocking	$2^{20.5}$	146 GB
[3] - Baised Birthday Attack	2^{42}	1 second	A5/1 Clocking	$2^{20.5}$	292 GB
[3] - Random Subgraph Attack	2^{48}	minutes	A5/1 Clocking	$2^{14.7}$	146 GB
Our Results	2^{38}	$2^{39.91}$	A5/1 Clocking	$2^{20.8}$	64 GB
Our Results	$2^{33.6}$	$2^{40.97}$	A5/1 Clocking	$2^{20.8}$	4 GB

Table 2: Attacks on A5/1 and their complexities

[2] Biryukov Alex, Shamir Adi, *Real Time cryptanalysis of A5/1*, private communication.

[3] Alex Biryukov, Adi Shamir, David Wagner, *Real Time Cryptanalysis of A5/1 on a PC*, Preproceedings of FSE '7, pp. 1-18, 2000.

[4] Marc Briceno, Ian Goldberg, David Wagner, *A Pedagogical Implementation of A5/1.*

[5] Marc Briceno, Ian Goldverg, David Wagner,
A Pedagogical Implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms. Available at *http://www.scard.org/gsm/a51.html.*

[6] Jovan Golic, *On the Security of Nonlinear Filter Generators*, Proceedings of Fast Software Encryption - FSE '96, Springer vol. 1039, pp. 173-188, 1996.

[7] Jovan Golic, *Cryptanalysis of Alleged A5 Stream Cipher*, Proceedings of Eurocrypt '97, Springer LNCS vol. 1233, pp. 239-255, 1997.