

KNOWN ATTACKS FOR THE A5/1 ALGORITHM: A TUTORIAL

Sherif Essam AlAschkar salaschkar@yahoo.com
Mahmoud Taher El-Hadidi hadidi@mailier.eun.eg
Electronics and Electrical Communications department, Cairo University, Egypt.

Abstract: This paper describes and compares some of the known attacks against the A5/1 algorithm, which is the strong version of the A5 encryption algorithms used in GSM to secure the air interface between the base stations and the mobile users.

Keywords: A5/1, GSM, Cryptanalysis, Stream cipher.

1 INTRODUCTION

The A5 algorithm is the protection provided by GSM for the over-the-air communication between the base stations and the mobile users. Among the A5 different algorithm versions, the A5/1 is the strongest one applied in commercial GSM systems till now. That is why the A5/1 is a good target for attackers to work on for breaking. If the algorithm is broken then the confidentiality of the user traffic is jeopardized. Many attacks have been conducted against the A5/1 algorithm. This paper explains the different attacks and compares between them.

In this paper, section 2 describes the A5/1 algorithm and its operation steps. Section 3 provides the details of the chosen attacks against the A5/1 algorithm pointing out the attack steps, requirements, advantages and disadvantages. Section 4 illustrates the differences between the different attacks based on the pre-computation complexity, the known plaintext/ciphertext amount, the memory complexity and the time complexity. Section 5 is the conclusion deduced from the study. Section 6 lists all of the used references.

2 A5/1 DESCRIPTION AND OPERATION

The A5 algorithm is the ciphering/deciphering algorithm used by GSM to secure the information sent over the air interface. The first original A5 algorithm was renamed to A5/1 and was allowed to be used by countries that are members of the Conference of European Post and Telecommunications (CEPT). Other countries are allowed to use either a weaker version, which is A5/2 or no encryption at all, which is referred to as A5/0, [10] or [4]. The version of the A5 algorithm that is studied here is A5/1. It is chosen since it is the strongest version of the A5 algorithm that is currently used in working GSM systems.

The A5/1 algorithm is presented in Fig. 2.1. It is a pseudorandom number generator that consists of three Maximal Length Linear Feedback Shift Registers (MLFSR). The three registers are of lengths 19, 22 and 23 and are denoted by R1, R2 and R3 respectively. The bit numbering and the feedback taps of the three registers are as shown in Fig. 2.1.

Each register has a single clocking bit, which is bit 8 for R1 and bit 10 for R2 and R3. A majority function is calculated from the clocking bit of each register. The function is equal to zero if at least two of the three bits are zero and one if at least two are one. Only the register whose clocking bit agrees with the majority function is clocked, this is called the stop/go rule. This clock control is the only non-linearity in the A5/1 algorithm design.

The registers are clocked by performing a left shift then updating the least significant bit by the value calculated from the feedback. The output of the algorithm is the XOR of the most significant bit of each register. The inputs to the algorithm are the 64 bit ciphering key (KC) and the publicly known 22 bits frame counter (COUNT), [1].

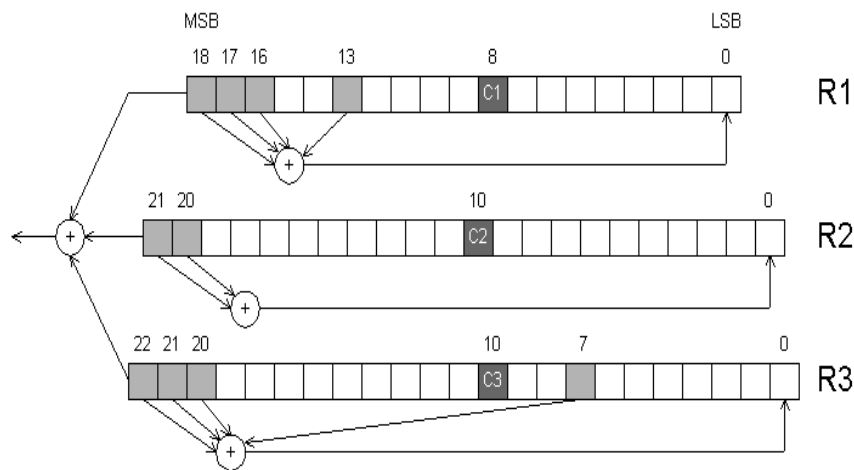


Fig. 2.1. The A5/1 algorithm

The operational steps of the A5/1 algorithm are as follows:

- **Initialization:** The bits of the registers are set to zero. The algorithm is in state S^0 .
- **Input K_C :** The registers are clocked with the stop/go rule disabled for 64 clock cycles. In each cycle, a bit from K_C , from its least significant to its most significant bits, is XORed into bit zero of each register. The algorithm reached state S^{64} .
- **Input COUNT:** It is the same as step 2, but it is 22 clock cycles long and COUNT is the input instead of K_C . The algorithm reached state S^{86} .
- **Mixing:** The registers are clocked for 100 mixing cycles using the stop/go rule. The output generated is ignored and the algorithm reached state S^{186} .
- **Output generation:** The registers are clocked for 228 times using the stop/go rule and the output generated is divided into 2 parts, where each part is 114 bits long. The first part is bit wise XORed with a frame from the received ciphertext so as to decrypt it while the second part is bit wise XORed with a frame from the plaintext that should be sent so as to encrypt it. The 228 bits are called the cipher stream. The algorithm reached state S^{414} at the end of this step.

- The steps from 1 to 5 are repeated with each new frame. Each time COUNT is incremented while K_C stays the same until authentication is triggered.

The states through which the A5/1 algorithm passes are shown in Fig. 2.2.

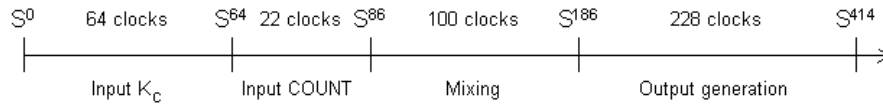


Fig. 2.2. State flow of the A5/1 algorithm

3 KNOWN ATTACKS

The attacks against A5/1 aim at generating the cipher streams that were generated by the algorithm. To do this, the attacker needs to know K_C and COUNT that were used with a certain frame. The value of COUNT could be derived from the frame number. So the problem is to get the value of K_C .

Instead of getting the value of K_C , the attacker can get the content of the registers right after the Input K_C stage, i.e. at S^{64} . This is because a certain value of K_C corresponds to a certain value of S^{64} , then knowing S^{64} is equivalent to knowing K_C . The attacker can run the algorithm starting from this state to get the cipher stream.

Most of the attacks against A5/1 are known plaintext attacks, where the attacker should know the plaintext and its equivalent ciphertext for as much frames as the attack requires.

The differences between the attacks against A5/1 are in the following attack parameters:

- Pre-computation complexity: The complexity of the preparation phase for an attack.
- Known plaintext/ciphertext: The amount of known plaintext and ciphertext pairs that are required by an attack. It could be in bits or seconds.
- Memory complexity: The storage required to carry out an attack.
- Time complexity: The complexity of carrying out an attack.

The attacks against A5/1 that are selected to be studied are referred to as follows:

- Golic: Cryptanalysis of the Alleged A5 Stream Cipher by Jovan Dj. Golic.
- Biryukov: Real Time Cryptanalysis of A5/1 on a PC by Alex Biryukov, Adi Shamir and David Wagner.
- Keller: A Hardware-Based Attack on the A5/1 Stream Cipher by Jörg Keller.
- Biham: Cryptanalysis of the A5/1 GSM Steam Cipher by Eli Biham and Orr Dunkelman.
- Krause: BDD-based Cryptanalysis of Keystream Generators by Matthias Krause.
- Ekdahl: Another Attack on A5/1 by Patrik Ekdahl and Thomas Johansson.
- Pornin: Software-Hardware Tradeoffs; application to A5/1 Cryptanalysis by Thomas Pornin and Jacques Stern.

The following are the details of selected attacks:

3.1 Cryptanalysis of the Alleged A5 Stream Cipher[7]

Two attacks on the A5/1 algorithm are introduced. One is a divide-and-conquer attack, which has an average computational complexity of $2^{40.16}$ and the other is a time-memory tradeoff attack based on the birthday paradox. The A5/1 algorithm used in the two attacks differ from the correct one in that it has a 100-clock cycles mixing step after the generation of the first 114 bits of the cipher stream.

a. The Divide-and-Conquer Attack

The divide-and-conquer attack aims at determining the initial state, S^0 , of the algorithm after the Input COUNT step. It needs only 64 known successive cipher stream bits.

The first step in the attack is to get S^{101} , which is the state after the Mixing step. First guess n bits for each LFSR starting from the bit at the clock control tap position. The guessing gives $3n$ bits. The $3n$ bits are used to form $3n$ linearly independent equations for the unknown bits of S^{101} provided that $n \leq 18$. On average, the $3n$ guessed bits provide $4n/3$ clock sequences. Therefore $1+4n/3$ additional equations are obtained, where the first equation is provided from the first stream cipher bit before clocking the algorithm. The

$1+4n/3$ equations are mutually linearly independent because each one contains 2 or more bits that have not been used before.

If n is taken equal to 10, then $1+3n+4n/3 = 44.3$ linear equations are obtained. So the remaining unspecified bits are $63.32 - 44.3 = 19.02$, which can be determined by reversion tree method. Each node in the tree is an internal state and nodes leaving it are the possible predecessors for that state. The average number of branches leaving a node is found to be 2.5. The depth of the tree should be $4m/3$, where m is the number of unknown bits in each LFSR that is approximately $19.02/3$. So instead of having $23m$ possibilities for the unknown bits it is reduced to $2.54m/3 = 211.16$. The complexity of the first part of the attack is $230+11.16 = 241.16$, which is 240.16 on average.

The second step in the attack is to get S^0 from S^{101} . This is done by guessing the number of clocks needed to get S_i^{101} from S_i^0 , where i is the number of the LFSR. The number of clocks is on average $4*101/3 = 76$. In each guess, S_i^0 is calculated from S_i^{101} by backward linear recursion then the guess is tested by running the algorithm forward.

b. The Time-Memory Tradeoff Attack

The objective of the attack is to find the preceding internal states for any successive 64 bits of the cipher stream and continue to get S^0 . The attack succeeds if $T*M \geq 2^{63.32}$, where T and M are the required computation time and memory respectively.

At most, all of the cipher streams that could be generated by a single K_C using the entire range of COUNT are assumed to be known, which is 2^{22} 228-bit cipher streams. In the general case only K values of COUNT are known, where $K \leq 2^{22}$. A known 228-bit cipher stream is divided into 2 separate 114-bit streams and each one is divided into 51 different 64-bit blocks. So the 228 bits of the cipher stream provide 102 64-bit blocks.

The idea is to form a lookup table with M entries that contains the 64-bit internal state of the algorithm and the output block that it generates. The internal states are randomly sampled from the available range. The table is sorted on the output blocks. The search in the table is based on the 102 output blocks. Searching the table yields a number of internal states that can give one of the known output blocks.

The intersection between the $102*K$ known blocks and the M output blocks stored in the table occurs if: $102*K*M \geq 2^{63.32}$. The time required to find a common block is $T = 102*K*\log(M) \approx 102*K$. Therefore the time-memory tradeoff is possible if $T*M \geq 2^{63.32}$ and only one table lookup is

needed to find the internal state. The time and memory requirements of the attack are shown by the following 2 examples:

- If $K = 15$, therefore $T = 102 * 2^{15} = 2^{21.67}$ and $M = 2^{41.65} = 55,210$ Gbytes
- If $K = 21$, therefore $T = 102 * 2^{21} = 2^{27.67}$ and $M = 2^{35.65} = 862$ Gbytes

The next step is to apply internal state reversion to get S^{101} then S^0 . Two internal state reversion steps are needed. One when the output is not known, i.e. in the mixing steps and one when the output is known, i.e. in the output generation steps.

Key Reconstruction

After getting S^0 using one of the two attacks, the final step is to get the state of the algorithm right after the Input K_C step knowing COUNT. First a bit of COUNT is added to the last bit of each LFSR and second, the LFSRs are clocked backwards using all of the possible combinations of the clocking bits and choosing only the combination that can give the state before the bit of COUNT is added. This is repeated to get S^0 .

Advantages

- The divide-and-conquer attack needs only 64 bits of known plaintext and ciphertext.
- The divide-and-conquer attack has no preprocessing stage and has no memory requirements.

Disadvantages

- The design of the A5/1 algorithm used in the attack is not fully correct.
- In the divide-and-conquer attack, each operation is based on the solution of a system of linear equations, which is very time consuming. The time complexity, which is $2^{40.16}$ in units of solution of linear equations, is equivalent to 2^{47} in units of A5/1 clocking, [2].
- The time-memory tradeoff attack has huge memory and time requirement, as it needs 862 GB and $2^{27.67}$ respectively.
- The time-memory tradeoff attack needs about 10 hours of known plaintext, which is impractical.
- In the time-memory tradeoff attack, Golic assumed that the critical branching process is a good statistical model for the generated trees of predecessors. This was proven wrong, so the statistical results obtained by Golic are not fully correct, [3].

3.2 Real Time Cryptanalysis of A5/1 on a PC

Two attacks on the A5/1 algorithm are introduced. The first one is the biased birthday attack and the second one is the random subgraph attack. Both attacks are based on the time-memory tradeoff introduced by Golic. Each of them starts by a preprocessing phase that prepares the data needed for the attack.

Preprocessing stage

In this stage, a large set “A” of pre-computed states is prepared and stored. For each stored state, it is needed to store the state itself and part of the output generated when running the A5/1 starting from this state, which is called the output prefix.

Only special states are kept on the disk. They are states that produce an output sequence starting with a particular pattern α (alpha) of length K, which is taken as the bit pattern 1000000000000000 (1 followed by 15 zeros). The number of states that can produce α is about $2^{64} * 2^{-16} = 2^{48}$, which should be generated without trying all of the 2^{64} states. This is performed by a number of steps that are proposed by the attackers. The attackers also proposed a method to efficiently store the states and their output prefixes in a (prefix, state) table ordered by the prefixes.

General Steps of the Attacks

Set “B” is defined, which contains the unknown states that the algorithm passes through during the generation of the known output bits. Each of the known 228-bit long outputs is searched for the occurrence of one of the known prefixes that are stored in the set “A”. This is done by searching for α in the known output and when an occurrence is found, the next 35 bits of the output are taken and searched for in the (prefix, state) table. The result of the search is a list of candidate states that fall in the part from S^{101} to S^{177} . These states are the outcome of both attacks.

The next step is to run the A5/1 backwards to find the corresponding S^1 candidate states. This is done by exploring the tree of possible predecessor states and backtracking from dead ends. The average number of predecessor for a state is 1 therefore the tree grows linearly with the number of backward steps. Finally, it is required to get the state of the A5/1 algorithm after the Input KC step from the S^1 state. For each of the S^1 states, the effect of COUNT can be eliminated in a unique way by running the algorithm backwards knowing its input, which is COUNT.

a. The Biased Birthday Attack

A state is called red if it generates output starting by α . The subspace of red states is R . Also a state is called green if it generates output that contains α i.e. α starts between bit 1 and bit 177 of the output. The subspace of green states is G .

The concept of the attack is to obtain a list of red states between S^{101} and S^{117} by colliding the stored data with the actual output. Then to get the green states that result from the known red states by reversing the A5/1 and then to proceed to get S^1 . The attack needs the data of the first 2 minutes of the conversation.

More than one green state can lead to the same red state at a certain point in time. This can be represented as a tree whose vertex is a red state and whose base is a belt of green states that are at levels from 101 to 277 below the root state. The weight $W(s)$ of a tree whose root is the red state s is defined as the number of states in the green belt of its tree.

In the attack, $|A|$ is 2^{35} , which is a 2^{-13} fraction of the available 2^{48} red states. If heavy tree are chosen for the disk, W_{avg} becomes 12,500 and the expected number of collisions becomes 0.61. This means that the attack is likely to succeed.

b. The Random Subgraph Attack

The idea of the attack is to use indirect collision between a state in the disk and a state in the real output of the algorithm. This makes the attack possible using the first red state found in the real output even if it is not stored in the disk, this state could be found in the first 2 seconds of a conversation. This is done by using the state not in the disk to find a state in the disk.

The attack is based on Hellman's time-memory tradeoff. Let E be an encryption function, P be the set of plaintexts, C be the set of ciphertexts and K be the set of keys. The function f is defined as $f(K) = E_K(P)$ and it is from K to C . If K , C and P have the same binary size, then f could be considered as a random function over a common space U .

The idea introduced by Hellman was to choose a large number, m , of start points from U , run f on each one of the points iteratively for t times and then store the m start and end points sorted by the end points. So if the value of $f(K)$ is given for an unknown K then f could be applied in the forward way repeatedly until a stored end point is found. The start point is easily found for the end point from the stored data then f is applied on the start point till $f(K)$

is found. The last point before finding $f(K)$ is K , which is equivalent to the known $f(K)$.

A re-randomization technique is proposed by Hellman. The idea is to create t variants of f and iterate each one of them t times on the m start points. This was proposed because it is difficult to cover a random graph with random paths efficiently.

For the A5/1 algorithm, the function f could be designed as the mapping between the 48-bit state representation and the 48-bit output prefix generated by the state after α is removed. This is done using the pre-processing stage by considering the 48 bits after α .

In the preprocessing stage, a red state start point is randomly selected and then iterated over the variants of f to jump from a red state to another. This process continues 2^{12} times till a bright red state is found, which is a state that gives in its output sequence the 16-bit α followed by 12 zeros, at which iteration stops and the start and end points are stored. The size of the stored end points in this implementation is reduced from 48 bits to 36 bits since the first 12 bits are always zeros.

During the attack using the bright red states, the data is searched for a red state, then each of the 2^{12} variants of f are iterated over it until a bright red state is found. The data stored in the disks are searched for the occurrence of the bright red state. Thus the disks are searched once in each of the 2^{12} tables, so the total time of accessing the disks becomes $6 \cdot 10^{-3} \cdot 2^{12} \approx 24$ seconds.

Advantages

- Both attacks are carried out in nearly real time on a PC.
- Both attacks require few frames of known plaintext/ciphertext pairs.

Disadvantages

- Both attacks have high memory and time complexities for their pre-processing stages.

3.3 A Hardware-Based Attack on the A5/1 Stream Cipher[8]

The attack is based on ideas from previous attacks, mainly from [3]. The state of the algorithm after the Input K_C step is S'' , after the Input COUNT step is S' and after the Mixing step is S . The attack is divided into three steps as follows:

a. Get S From the Known Output

The first step is to get all of the states, S , that can give the 64 known bits of the output. This is the time-consuming part of the attack, so it is implemented in field programmable gate array (FPGA).

The idea of this step is to construct 2^{41} partial states, at position S , by choosing arbitrary values for the 19 bits of $R1$ and for the 22 bits of $R2$. Then compute all states that $R3$ could have had at position S , to give the first 64 bits of the known output sequence. Then at the end, test that the full state at S can generate the known output sequence.

The author proposed to implement this step on 1000 large ASICs in 0.1μ technology. The 2^{41} checks would need about 70 sec. On average the time needed is about 35 sec, which is nearly real time. The cost of such a machine connected to a normal PC is from 1 to 2 million US dollars.

b. Get S' from S

The second step is to generate from S the state S' that is before the Mixing step. In this step, the majority clocking rule is applied so there could be up to 4 predecessor states for each state. However, it was found that the average number of predecessor states is 1, so the number of possible states in the tree of predecessor states grows only linearly with the level of the tree. This makes the backtracking from state S to state S' possible.

c. Get S'' from S'

The third step is to computing the state S'' that corresponds to each of the S' states that are found in the previous step. This is possible since the three registers are clocked in the Input COUNT step and the value of COUNT is known. This makes the backtracking from state S' to state S'' possible. It is expected that only 1 state S'' will be reached, with high priority, from the 64 known output bits.

Advantages

- The attack needs only 64 bits of known plaintext/ciphertext.
- A distributed implementation of the attack on 1000 ASICs could recover a session key in less than a minute.
- The attack does not use pre-computed large tables that have high memory requirements.

Disadvantages

- The cost of the attack is very high, which was estimated between 1 and 2 million US dollars.

3.4 Cryptanalysis of the A5/1 GSM Stream Cipher[2]

The main idea of the attack is to wait until an event, which leaks a large amount of information about the key occurs and then exploit it. The event used is that for 10 consecutive clock cycles, R3 is not clocked while R1 and R2 are clocked.

The bits of the registers are numbered starting from 0 at the input bit of the register to the length of the register minus 1 at its output bit. $RX[Y]$ is bit Y in register X. The numbering used is based on the state of the algorithm after the mixing step.

The steps of the attack are as follows:

- Choose arbitrary values for R3[10], which is the clocking bit and R3[22], which is the output bit of the register. Therefore for the next 10 clock cycles, where R3 will not be clocked, the clock control bits in R1 and R2 are the complement of R3[10]. Therefore, $R1[i]=R2[j]=R3[10]$, where $i:0 \dots 8, j: 1 \dots 10$ and the first bit that is shifted into R1 from the feedback is R3[10]. R3[22] will contribute in the value of at least 11 output bits, this gives the following 11 equations:(Respective output bit)

$$\text{xor } R3[22] = R1[k] \text{ xor } R2[k+3], \text{ where } k: 8 \dots 18.$$

- Choose 9 arbitrary bits from R1, which are R1[l], where $l: 9 \dots 18$ and 1 bit from R2, which is R2[0]. Thus all of the bits of R1 and R2 are specified. R1[13] should not be guessed because the feedback of R1 in the first clock cycle is equal to R3[10]. Therefore, guessing R1[18], R1[17] and R1[16] is enough to determine R1[13] from the equation:

$$R1[13] = R1[18] \text{ xor } R1[17] \text{ xor } R1[16] \text{ xor } R3[10]$$

- R2[11] can be computed knowing the output stream from the equation:

$$R1[8] \text{ xor } R2[11] = R3[22] \text{ xor (Respective output bit), where } R1[8] \text{ is known as it is equal to } R3[10].$$

- The algorithm is clocked till R3 moves so as to calculate R3[21] from the output and the known bits of R1 and R2.
- Choose arbitrary values for R3[m], where m: 0 ... 9 and continue clocking the algorithm to calculate R3[n], where n: 11 ... 20, which are 11 bits. This step needs that R3 clocks for 12 clock cycles. This needs 16 clock cycles of the A5/1 on average.
- At this point, the value of the bits of the 3 registers is determined, so to check the correctness of the internal state 2 clock cycles are needed for each guess.

The total expected running time of the attack is found to be 227 knowing the location where R3 is not moving for 10 clock cycles. Since this location is not known, then it is required to examine about 220 starting locations to find it. Therefore the time complexity of the attack is 247 and it requires 220 known output bits.

The attackers proposed some methods to reduce the time complexity of the attack as follows:

1. Two tables are generated per register. The first one, the next-state table, contains the states that each register passes through when clocked and the second one, pointers table gives for each state x , its location in the next state table. Using the 2 tables, per register, a state can be clocked for any number of clock cycles in a fixed time. The memory needed for the 6 tables is about 71.5 MB.
2. A technique is proposed to reduce the number of possible values of the R3 bits that generate the known output. This is based on a new table. The table contains the output generated by the 5 most significant bits in each register and the next 5 clock controlling bits. It is indexed by the output. The table contains information on how many clock cycles each register needs to be clocked after the output has been generated by the 30 bits (10 from each register). The generation of the table requires $2^{32.6}$ A5/1 clock cycles and needs 2 GB.

Using the modifications, the complexity of the attack becomes $2^{40.97}$ A5/1 clock cycles and could be improved to become $2^{39.91}$ by storing 12 bits from R1 and R2 instead of 10 in the table. In this case, the pre-computation complexity becomes 2^{37} A5/1 clock cycles and the table needs about 32 GB.

Advantages

- The memory requirements of the attack is relatively small (32 GB or 2 GB).
- The pre-computation complexity is relatively smaller than other attacks.

Disadvantages

- The attack is based on a special event, so the attacker has to wait till this event occurs so as to use it in the attack.

3.5 BDD-Based Cryptanalysis of Keystream Generators [9]

The attack is based on Free Binary Decision Diagrams (FBDDs), introduced by J. Gergov and Ch. Meinel in [6] and by D. Sieling and I. Wegener in [12]. The objective of the attack is to find the secret initial state of the algorithm knowing a number of output frames. The problem of finding the initial state can be reduced to finding the minimal FBDD, P , for the decision if the initial state can give the output stream.

The attack is based on a weakness in the output generation method of the A5/1, as it is the XOR of the most significant bit of the registers, so it is memoryless. Therefore, a sequence of small FBDDs, P_m , where $m \geq n$, can be dynamically computed and used to decide whether an internal bitstream can generate a prefix of a given output stream. The symbol m is the output prefix and n is the number of bits in the A5/1, which is 64.

The FBDD gives a unique solution for the initial states if $m \geq \lceil \alpha^{-1}n \rceil$, where α is the rate of information revealed about the internal bit stream. The FBDDs need the first $\lceil \gamma m \rceil$ bits of the cipher stream, where γ is the compression ratio of the output. Thus the attack needs the first $\lceil \gamma \alpha^{-1}n \rceil$ bits of the output.

Nearly all of the internal bits of the A5/1 are used twice, once to compute the majority clocking rule and once in output generation. Read-twice BDDs do not have good properties. Therefore, Krause modified the A5/1 to contain 6 LFSRs instead of 3. The output is generated from the first 3 LFSRs while the clock control from the other 3.

For the modified algorithm, $\gamma = 1/4$ and $\alpha = 0.2193$. Therefore, the time complexity of the FBDD attack is $2^{\frac{1-\alpha}{1+\alpha}n} = 2^{\frac{1-0.2193}{1+0.2193}64} = 2^{41}$ and the number of bits of known plaintext/cipher needed is $\lceil \gamma m \rceil = \lceil \gamma \alpha^{-1}n \rceil = \lceil \lceil \rceil \rceil 73$ bits.

Advantages

- The FBDD attack has a better time behavior than all other attacks, not comparing the results to the results of time-memory trade-offs, which could be better.
- The attack does not need a long pre-processing stage and it does not have memory requirements.

Disadvantages

- The time complexity of the attack is greater than the time complexity of time-memory tradeoff attacks.

3.6 Another Attack on A5/1[5]

The attack is based on the bad key initialization of the A5/1, where K_C and COUNT are initialized in a linear fashion, which allows separating them in a binary linear expression.

The following are the terms used in the attack:

- u_i^j : Output bit number i of the shift register number j starting from the initial state.
- F_n : Value of COUNT, where $F_n = (f_1, f_2, \dots, f_{22})$.
- K : Ciphering Key (K_C), where $K = (k_1, k_2, \dots, k_{64})$.
- Z : Cipher Stream, where $Z = (z_1, z_2, \dots, z_{228})$.

The initial state is found to be a linear function of K and F_n . The output sequence of an LFSR is $u_t^1 = \sum_{i=1}^{64} c_{\#}^1 \cdot k_i + \sum_{i=1}^{22} d_{\#}^1 \cdot f_i$, where t is the output bit number that is ≥ 0 , 1 is the LFSR number and c_{it}^1 and d_{it}^1 are binary constants.

Let $s_t^1 = \sum_{i=1}^{64} c_{\#}^1 \cdot k_i$ and $\hat{f}_t^1 = \sum_{i=1}^{22} d_{\#}^1 \cdot f_i$. Therefore, $u_t^1 = s_t^1 + \hat{f}_t^1$, where s_t^1 is the key part of U_t^1 and is an unknown binary sequence that is the same for all frames and \hat{f}_t^1 is the frame number part of U_t^1 and is a known binary sequence that differs for each value of F_n and is known because F_n is known.

The first cipher stream bit, z_1 , is generated after 101 clock cycles. Each LFSR moves on average $\frac{3}{4}$ of the time, which is 76 clock cycles, therefore, since \hat{f}_t^1 is known, therefore using $u_t^1 = s_t^1 + \hat{f}_t^1$, gives:

$$s_{76}^1 + s_{76}^2 + s_{76}^3 = \hat{f}_{76}^1 + \hat{f}_{76}^2 + \hat{f}_{76}^3 + z_1 \quad (1)$$

The right hand side of equation (1) contains known quantities so it is represented by $O_{(76,:)}^j$, where j is the frame number.

The probability that the 3 registers are clocked 76 times is 10^{-3} . If this assumption is wrong, therefore equation (1) holds but with probability $\frac{1}{2}$. Therefore a correlation is identified by calculating

$$P(s_{76}^1 + s_{76}^2 + s_{76}^3 = O_{(76,76,76,1)}^j) = \frac{1}{2} + \frac{1}{2} \cdot 10^{-3}.$$

The left hand side, $s_{76}^1 + s_{76}^2 + s_{76}^3$, is constant for all frames, so if the attacker has access to few million frames then he can calculate $O_{(76,:)}^j$ for each frame and can determine $s_{76}^1 + s_{76}^2 + s_{76}^3$.

The authors proposed a method for decreasing the number of needed known frames and making the attack faster. The clocking (cl_1, cl_2, cl_3) might end up in positions other than z_1 , like z_2, \dots, z_{228} . So the correlation probability in frame number j is defined as

$P_{(cl_1,cl_2,cl_3)}^j = P(s_{cl_1}^1 + s_{cl_2}^2 + s_{cl_3}^3 = 0)$ by averaging over all positions from 1 to 228.

Let $E\{k\}$ be the event that (cl_1, cl_2, cl_3) is in position k . Therefore,

$$P_{(cl_1,cl_2,cl_3)}^j = \sum_{k=1}^{228} P(E\{k\}) \cdot [O_{(cl_1,cl_2,cl_3,v-100)}^j = 0] + \frac{1}{2} \cdot (1 - \sum_{k=1}^{228} P(E\{k\}))$$

$P_{(cl_1,cl)}^j$ is averaged over all of the frames using a log likelihood function to make a soft decision on the value of $P(s_{cl_1}^1 + s_{cl_2}^2 + s_{cl_3}^3 = 0)$. the key can then be calculated from the estimates of the different (cl_1, cl_2, cl_3).

The attack needs a pre-computation phase, that requires about 15 minutes of calculation time and less than 2 MB of storage. The attack needs from 2 to 5 minutes of known plaintext/ciphertext.

Advantages

- The complexity of the attack is almost independent of the length of the shift registers.
- The attack needs less than 5 minutes on a normal PC to break the A5/1 algorithm.
- The attack does not need huge storage, as the memory requirement is only 2 MB.
- The attack has a small pre-computation stage, which needs 15 minutes on a PC.
- The success rate of the attack is more than 70%.

Disadvantages

- The attack needs a relatively large number of known plaintext/ciphertext frames to get the key.

3.7 Software-Hardware Trade-offs; Application to A5/1 Cryptanalysis [11]

The attack is based on a trade-off between a generic workstation and a fast re-configurable hardware to make use of the capabilities of both.

3.7.1 Software Alone Implementation

The attack is a modified version of the divide-and-conquer attack by Golic, in [7]. It is based on guessing the clock sequence for a given number of clock cycles. Each output bit is a linear equation of the unknown internal state bits. The system is then reversed and the initial state is recovered. The modified version of the Golic attack is based on backtracking in a predecessor tree. Each guess in the backtracking process provides 3 equations that are linear with 64 unknowns that are the 64 bit initial state.

The average complexity of the attack is $2^{44.3}$, where each step is the Gaussian Elimination of one equation. The implementation takes on average about 400 days on a Compaq XP-1000 workstation using a 21264 Alpha processor at 500 MHz.

3.7.2 Hardware Alone Implementation

It is implemented using a Pamette, which is a PCI card that has a number of Xilinx 4010E FPGA implementations. The characteristics of the implementation are as follows:

- In each clock cycle, 1 step of A5/1 is performed.
- It is possible to reload the registers with new values in 1 cycle.
- The implementation can run at 50 MHz.
- The Pamette can hold up to 48 instances of the A5/1.

The Pamette can try up to 37 million initial states per second. So an exhaustive search attack to get the 64-bit internal state needs 15,800 years on 1 Pamette.

3.7.3 Software-Hardware Tradeoff

A trade-off is proposed to make part of the attack with software and part with hardware. The performance is much better than the software alone and hardware alone implementations. When using 2 Pamettes per PC, the attack needs 2.5 days on average.

Advantages

- The attack needs only 64 bits of known plaintext/ciphertext.

Disadvantages

- The time complexity of the attack is much more than other attacks and it is not a real time attack.

4 COMPARISON BETWEEN THE DIFFERENT ATTACKS

The attacks are compared using the attack parameters. Each attack is referred to using the attack reference name followed by a number. The comparison is shown in Table 4.1.

Table 4.1. Comparison between the different attacks against the A5/1 algorithm

Ref. Name	Attack Name	Pre-comp Complexity	Known Plaintext/Ciphertext	Memory Complexity	Time Complexity
Golic1	Divide-and-Conquer	0	64 bits	0	2^{47}
Golic2	Time-Memory Tradeoff	$2^{35.65}$	$2^{28.8}$ bits	862 GB	$2^{27.67}$
Biryukov1	Biased Birthday	2^{48}	$2^{20.5}$ bits	146 GB	1 second ($\cong 0$)*
Biryukov2	Biased Birthday	2^{42}	$2^{20.5}$ bits	292 GB	1 second ($\cong 0$)*
Biryukov3	Random Subgraph	2^{48}	$2^{14.7}$ bits	146 GB	Few minutes ($\cong 0$)*
Keller	Hardware-Based Attack	0	64 bits	0	< 1 minute ($\cong 0$)*
Biham1	Cryptanalysis of A5/1	2^{38}	$2^{20.8}$ bits	32 GB	$2^{39.91}$
Biham2	Cryptanalysis of A5/1	$2^{33.6}$	$2^{20.8}$ bits	2 GB	$2^{40.97}$
Krause	BDD-based Cryptanalysis	0	73 bits	0	2^{41}
Ekdahl	Another Attack	15 minutes ($\cong 0$)*	$2^{23.83}$ bits	2 MB ($\cong 0$)*	< 5 minutes
Pornin	Software-Hardware Trade-offs	0	64 bits	0	2.5 days ($\cong 2^{41}$)* **

* The value between brackets is approximated relative to the values of other attacks.

** The calculation is based on the assumption that a normal PC can test 10 million guesses per second.

For each of the four parameters, a boundary value is chosen as follows:

- Pre-computation complexity: 2^{50} . It is allowed to be relatively high as the pre-computation step is carried only once.
- Known plaintext/ciphertext: 2^{24} , which is approximately 5 minutes because it is not likely that a conversation exceeds it and because this data is not easy to get.
- Memory complexity: 500 GB, which is a small number of hard disk drives.
- Time complexity: A few minutes. The attacks that are within this boundary could be considered real time attacks. Future technologies

might turn infeasible attacks into feasible ones due to higher processing powers.

Fig. 4.1, 4.2, 4.3 and 4.4 show the comparison between the attacks based on the pre-computation complexity, the known plaintext/ciphertext, the memory complexity and the time complexity respectively.

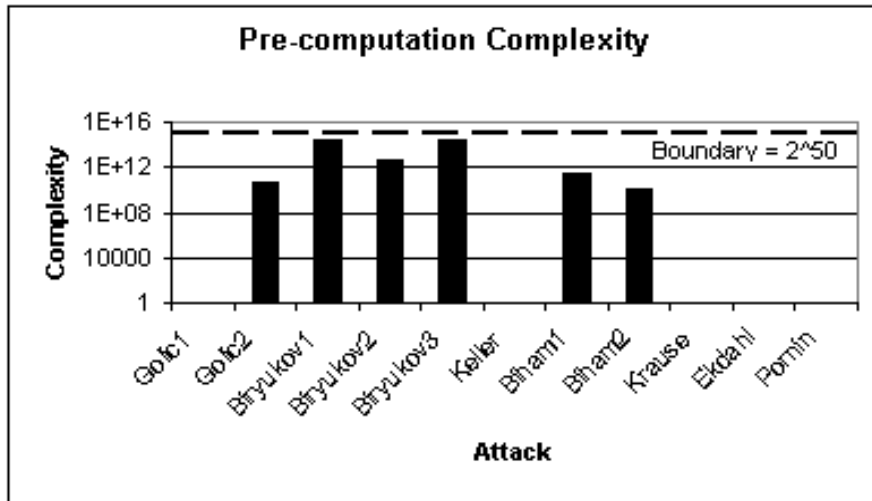


Fig. 4.1. Comparison between the attacks based on pre-computation complexity

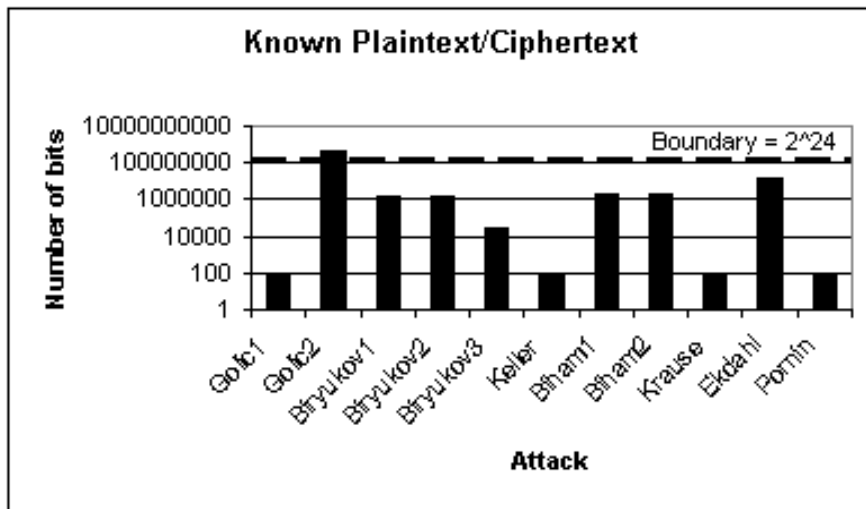


Fig. 4.2. Comparison between the attacks based on known plaintext/ciphertext

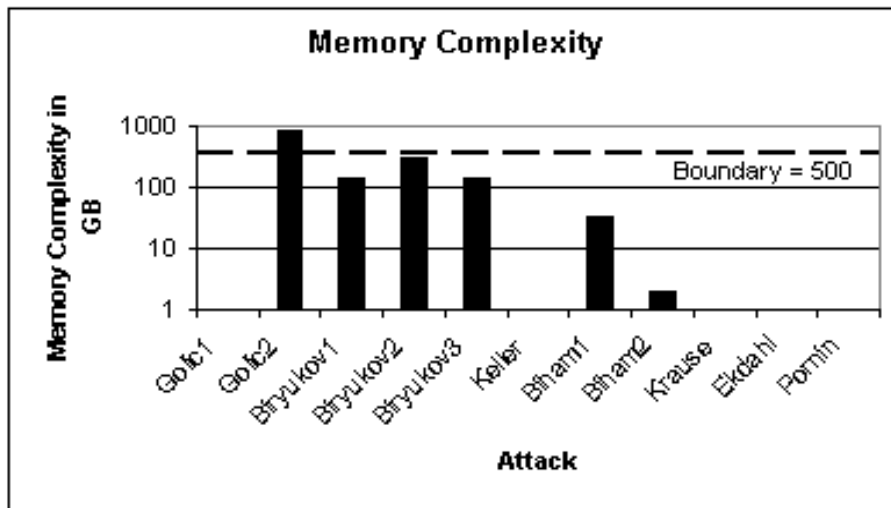


Fig. 4.3. Comparison between the attacks based on memory complexity

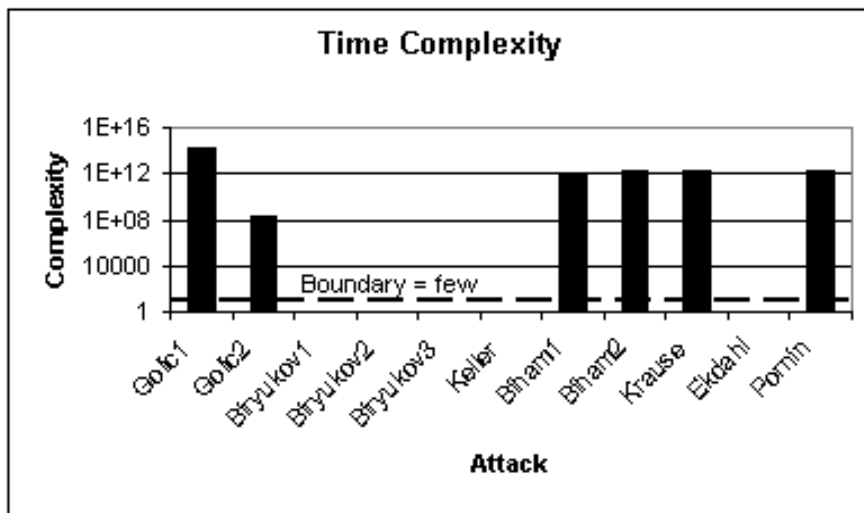


Fig. 4.4. Comparison between the attacks based on time complexity

The attack parameters are ordered in descending order of their importance i.e. in the order at which the parameters are to be considered. The most important parameter is the known plaintext/ciphertext, then the time complexity of the attack, then the memory complexity and finally the pre-computation complexity. Using this order and considering only the attacks

that have parameters below the boundary, filtering of the attacks is as follows:

- Known plaintext/ciphertext: The attacks that require less than 5 minutes (2^{24} bits) are: Golic1, Biryukov1, Biryukov2, Biryukov3, Keller, Biham1, Biham2, Krause, Pornin and Ekdahl.
- Time Complexity: From the attacks in step 1, the attacks that require a few minutes are chosen, which are: Biryukov1, Biryukov2, Biryukov3, Keller and Ekdahl.
- Memory Complexity: From the attacks in step 2, the attacks that require less than 500 GB are chosen, which are: Biryukov1, Biryukov2, Biryukov3, Keller and Ekdahl.
- Pre-computation Complexity: from the attacks in step 3, the attacks that require less than 2^{50} are chosen, which are: Biryukov1, Biryukov2, Biryukov3, Keller and Ekdahl.

The result of the filtering is that the Biryukov, Keller, and Ekdahl attacks are chosen. Each of these attacks has its own advantages and disadvantages, which might favor one of them over the others.

5 CONCLUSION

In this study, seven of the different attacks against the A5/1 algorithm are discussed in details. The attacks are known plaintext attacks. The discussion pointed out the attack details, advantages, disadvantages, pre-computation complexity, time complexity, memory requirements and amount of known plaintext/ciphertext pairs needed by each attack.

It is found that some of the attacks are software-only attacks while others use a mixture between software and hardware. Also some of the attacks are time-memory tradeoff attacks while others are based on the solution of linear equations or doing a lot of computations.

A comparison is made between the seven attacks based on the different attack parameters. The comparison pointed out that some of the attacks are infeasible, as they require a lot of known plaintext/ciphertext, processing time or storage. It also pointed out that some attacks could be considered as real time attacks against the A5/1 algorithm.

6 REFERENCES

- [1] Anonymous source (1997), "GSM System Security Study", RACAL Research ltd, <http://jya.com/gsm061088.htm>.
- [2] Biham E., Dunkelman O., "Cryptanalysis of the A5/1 GSM Stream Cipher".
- [3] Biryukov A., Shamir A., Wagner D.(April 10-12, 2000), "Real Time Cryptanalysis of A5/1 on a PC", Fast Encryption Workshop 2000, <http://www.counterpane.com/fse.html>, New York City.
- [4] Brookson C.(1994), "GSM (and PCN) Security and Encryption".
- [5] Ekdahl P., Johansson T., "Another Attack on A5/1".
- [6] Gergov J., Meinel C.(1994), "Efficient Boolean function manipulation with OBDDs can be generalized to FBDDs". IEEE Trans. on Computers 43, 1197-1209.
- [7] Golic J.(May 11-15, 1997), "Cryptanalysis of Alleged A5 Stream Cipher", Advances in Cryptography, EUROCRYPT 97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, 239-255, , <http://jya.com/a5-hack.htm>.
- [8] Keller J., "A Hardware-Based Attack on the A5/1 Stream Cipher".
- [9] Krause M., "BDD-based Cryptanalysis of Keystream generators".
- [10] Margrave D., "GSM Security and Encryption", George Mason University, <http://spyhard.narod.ru/phreak/gsm-secr.html>.
- [11] Pornin T., Stern J., "Software-Hardware Trade-offs; application to A5/1 Cryptanalysis".
- [12] Sieling D., Wegener I.(1995), "Graph driven BDDs – a new data structure for Boolean functions". Theoretical Computer Science 141, 283-310.