Tik-110.551: Attacks against A5

Lauri Tarkkala
ltarkkal@cc.hut.fi

Abstract

This paper describes the GSM voice confidentiality cipher A5/1 and surveys a set of attacks against it. The attacks clearly demonstrate that A5/1 does not provide sufficient levels of security for protecting the confidentiality of GSM voice traffic.

1 Introduction

GSM is a mobile communications system. The voice communication between the base station and the mobile handset is encrypted using a stream cipher. The encryption and decryption are performed by the handset and the base station of the network.

The cipher used for voice communication confidentiality is commonly known as A5/1, which is used in Europe. A weaker version called the A5/2 also exists, but it provides security against attackers capable of significantly less than 2^{16} operations. The algorithms were intended to be kept secret, but have been reverse engineered [1] based on information released in [4]. This paper considers attacks only against the cipher A5/1. A5/1 is a synchronous symmetric-key stream cipher, which relies on a 64-bit secret key.

2 GSM Voice Confidentiality

Every 4.6 ms of voice communication is digitized, compressed and divided into 114-bit frames of plaintext [6]. One second of voice communication contains approximately 217 frames uplink and downlink. Associated with each pair of downlink and uplink frames is a frame counter F_n. F_n is a 22-bit positive integer. The frame counter is disclosed during communication. [4]

The A5/1 algorithm is a pseudo-random generator $A5/1 : \{0,1\}^{64} \times \{0,1\}^{22} \rightarrow \{0,1\}^{228}$. The algorithm is input with a session secret $K_c \in \{0,1\}^{64}$ and the $F_n \in \{0,1\}^{22}$. Encryption and decryption are performed by XOR:ng the plaintext with these pseudo-random bits. The handset and base station (BS) use the first and last halves alternatively to encrypt and decrypt.

K_c does not change very often (in fact it need change only when the network authenticates the subscriber), but incrementing F_n for every invocation causes a different pseudo-random sequence to be generated for each frame.

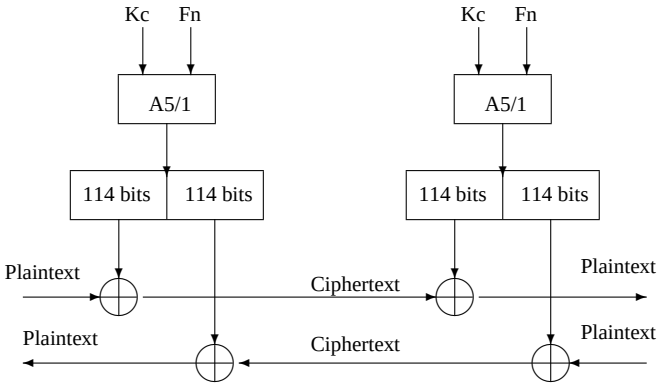


Figure 1: Encryption and Decryption using A5/1 for GSM

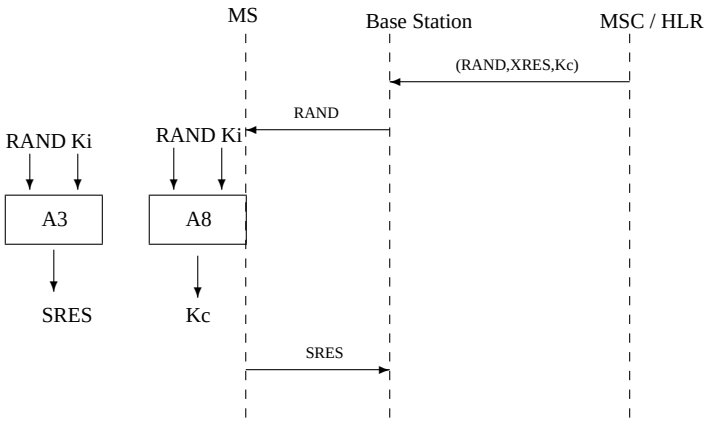


Figure 2: GSM subscriber authentication and session key agreement

The subscriber and the mobile switching center (MSC) in the fixed network share a 128-bit secret K_i, which is used to authenticate the mobile station to the network and for creation of the session secret K_c [5]. The key agreement protocol is depicted in Figure 2.

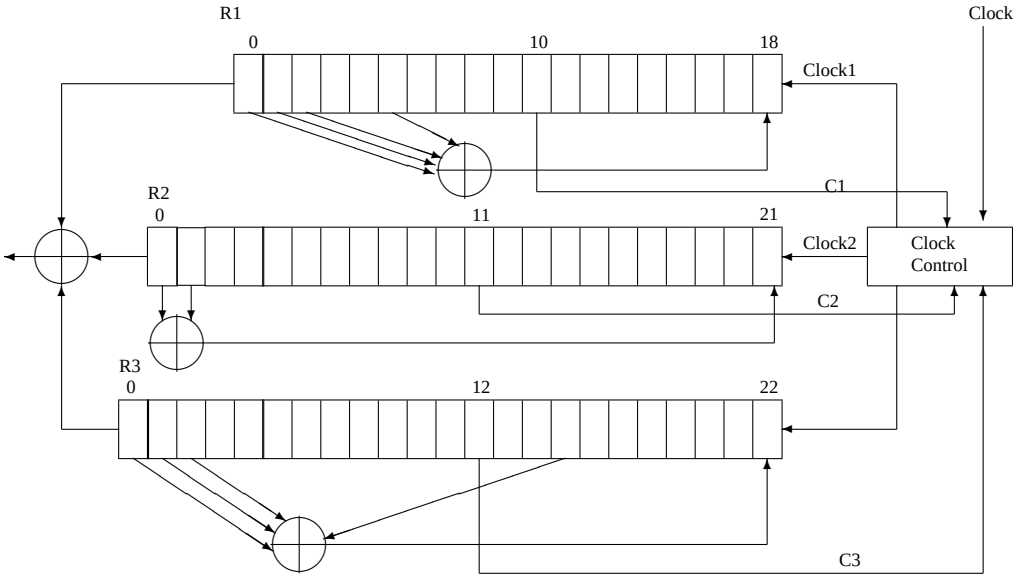
The A3 and A8 are both assumed to be one-way functions, otherwise the permanent secret K_i would be revealed. In practice the A3 and A8 algorithms are often based on the standardized $COMP128 : \{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ function.

The 32 most significant bits of the COMP128 output are used as the output of A3 (SRES). The least-significant output bits of the COMP128 output concatenated with 10 zero-bits are used as the output for A8 (K_c).

The effective key size for the A5/1 in practice may be assumed to be only 54-bits.

3 A.5/1 Structure

The A5/1 pseudo-random generator is constructed from three LSFR registers R1, R2 and R3. R1, R2 and R3 are 19, 22 and 23 bits in length and have periods $2^{19} - 1$, $2^{21} - 1$ and $2^{22} - 1$ respectively. The combination function is a XOR. The only non-linear component is the clock control mechanism.



$$clock1 = clock \wedge ((C1 \wedge (C2 \vee C3)) \vee \neg(C1 \vee (C2 \wedge C3)))$$

$$clock2 = clock \wedge ((C2 \wedge (C1 \vee C3)) \vee \neg(C2 \vee (C1 \wedge C3)))$$

$$clock3 = clock \wedge ((C3 \wedge (C1 \vee C2)) \vee \neg(C3 \vee (C1 \wedge C2)))$$

Figure 3: A5/1 structure

Denote by r_i the length of each register in bits.

Denote by $s_{i,l}(t)$ the state of bit l in register i at cycle t. The bits of the register are labeled from 0 to $r_i - 1$. Bit 0 is output bit.

Denote by $S_i(t)$ the state of register i at cycle t.

Denote by $S(t)$ the state of A5/1 at cycle t.

Denote by $y_i(t)$ the output bit of register i at clock cycle t. The first output bit of an LSFR is $y_i(1)$.

Denote by $y(t) = (y_1(t) + y_2(t) + y_3(t)) \; mod \; 2$ the output bit of A5/1 at cycle t.

Denote by τ_i the clock control bits from each register. $\tau_1 = 10$, $\tau_2 = 11$ and $\tau_3 = 12$. A LSFR is clocked only if it's input into the clock control is the same as the input from another LSFR.

The output of A5/1, the pseudo-random sequence $A5/1(K_c, F_n)$, is computed as follows.

1. Zero every bit in R1,R2 and R3
2. For each bit in K_c XOR the value of that bit into bit r_i of each LSFR and clock it

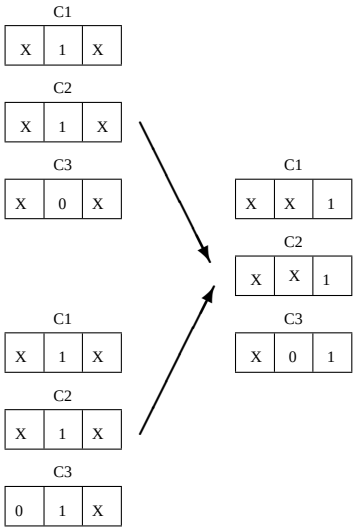


Figure 4: Simple example that A5/1 state transition is not injective

Amount of Predecessors	Probability
0	3/8
1	13/32
2	3/32
3	3/32
4	1/32

Table 1: State transitions function characteristics

(the "clock control" is ignored). The bits of K_c are traversed from lsb to msb.

3. For each bit in F_n XOR the value of that bit into bit r_i of each LSFR and clock it (the "clock control" is ignored). The bits of F_n are traversed from lsb to msb. The state of the registers after this step is called the *initial state*.

4. Clock the generator for 100 cycles using the clock control. Discard the 100 output bits.

5. Clock the generator for 114 cycles using the clock control. For each clock cycle one output bit is produced as the XOR of the outputs of R1, R2 and R3.

6. Clock the generator for 114 cycles using the clock control. For each clock cycle one output bit is produced as the XOR of the outputs of R1, R2 and R3.

Interestingly the specification of the algorithm above differs in [1] and [3]. The algorithm above is the one presented in [1].

The state transition function of A5/1 is not injective when the clock control is enabled. The amount of reachable states from the initial state is therefore less than the 2^{64} initial states. An example is shown in figure 4. The amount of states reachable is $5 * 2^{61} \approx 2^{63.32}$. Table 1 shows the distribution of predecessor states for a randomly chosen state. The expected number of predecessors is 1. [3]

(x, y, z)	$A_1(i+10)$	$A_2(j+11)$	$A_3(k+12)$
(1,1,1)	0	0	0
(1,1,0)	0	0	1
(1,0,1)	0	1	0
(0,1,1)	0	1	1
(0,1,1)	1	0	0
(1,0,1)	1	0	1
(1,1,0)	1	1	0
(1,1,1)	1	1	1

Table 2: Table lookup for A5/1 state transition function

3.1 Implementing A5/1 in software

An efficient software implementation if A5/1 is proposed in [2]. The implementation is based on the relatively short cycles of the three LSFR's R1, R2 and R3. As the three LSFR's all produce maximum length cycles, there is only one cycle per register of length $2^{r_i} - 1$.

Calculate the cycles produced by each of the registers and store them in memory. A state of an LSFR can now be represented using a triple which contains three indices into the computed sequences. If $A_j(i)$ denotes bit 0 of register Rj then $A_1(i+10), A_2(i+11)$ and $A(i+12)$ denote the clock control bits of the LSFR's. A state of A5/1 can now be represented as a simple triple $S(t) = (i, j, k)$. The state of R1 would then be $S_1(t) = A_1(0)A_1(i+1)A(i+2)...A(i+18)$.

The successor state $S(t+1)$ for $S(t) = (i, j, k)$ can now easily be computed using a table lookup into Table 2 as $S(t+1) = ((i+x)\ mod\ (2^{18}-1), (j+y)\ mod\ (2^{22}-1), (k+z)\ mod\ (2^{23}-1))$.

Calculating an output bit of A5/1 can now be performed using 3 table lookups and a X-OR. Calculating a state transition of A5/1 can be performed using 3 table lookups and 6 additions.

The total memory required by the three bit-sequences is approximately $(2^{19} + 2^{22} + 2^{23})/2^3 = 2^{16} + 2^{19} + 2^{20} < 2^{21}$ bytes or under 2 MB.

Table 2 can be expanded to consider several state transitions in one lookup for additional speed in "fast forwarding" A5/1. A table considering at least 8 transitions in one lookup would require 2^{24} rows, with each row containing at most 6 bytes. This table would require $6 \times 2^{24} < 2^3 \times 2^{24}$ bytes or under 128MB.

Similar table-lookup techniques can be used to run A5/1 backwards, although one must consider the possibility that there can be up to 4 preceding states for each state when constructing the tables.

4 Known Ciphertext Attack

Given a sequence of frame counters $F_n \in \{0,1\}^{22}$ and a sequence of pairs of ciphertext frames $C_n \in \{0,1\}^{114} \times \{0,1\}^{114}$ a known ciphertext attack against A5/1 aims to discover information about either the plaintext $P_n \in \{0,1\}^{114} \times \{0,1\}^{114}$ or the session key K_c.

A brute force attack iterates over the possible keys $K_c \in \{0,1\}^{64}$ and creates a set of pseudo-random sequences $X_n \in \{0,1\}^{114} \times \{0,1\}^{114}$ for each frame and computes the plaintext pairs P_n for each ciphertext pair C_n.

If the sequence of computed P_n's are all recognized as a valid plaintext, a candidate key K_c has been found. The feasibility of a brute-force attack in practice is limited partly by the possibility of efficient plaintext recognition.

Brute-force attacks have been performed against DES which is a symmetric block-cipher with a 56 bit key. [7] The internal structure of A5/1 is much simpler than that of DES and in practice the key space is 1/4 of that of DES.

According to the above software implementation, an A5/1 operation would require 6 table lookups and 6 additions per output bit. Generation of the 228 output bits from an initial state can easily be performed using only $300+228*6 = 1668$ table lookups and 1668 additions. It can be approximated that the total procedure of mixing the session key and frame counter and generating the 228 output bits can be performed in under 5000 operations.

Estimating on the basis of this that a PC could try approximately in between 10^5 and 10^7 keys per second. Searching the key space of size 2^{54} would require at least $(2^{54}/10^7)/(60*60*24) \approx 20000$ days. A network of 100 PC's may be able to crack a key in weeks.

The DES cracker machine tries a DES key in 16 clock cycles and the chips run at 40 Mhz with each chip containing 24 search units. The machine had a total of 36864 search units. To exhaust a key space of size 2^{54} this machine would require $2^{54}/(36864*40*10^6/16) \approx 195470$ seconds $= 5.4$ hours. The A5/1 algorithm is much simpler than DES and the clock speed and the density of search units on the chips could probably be increased significantly resulting in increased throughput. [7]

The problem remains in implementing an intelligent plaintext recognizer. A full-rate speech frame is encoded as 4 114-bit blocks in GSM, which represent only 18.4 ms of communication. The content of the frames is compressed, so it is safe to assume there is relatively little redundancy in the frames. The data is transported alongside with a simple CRC checksum. The CRC polynomial and it's use varies along with the type of data being handled. [6]

Assuming one iterates over 2^{54} possible keys and there are 32 redundant bits per 114-bit frame, the expected amount of keys which generate the correct redundancy is 2^{22}. The probability of a key key producing the correct redundancy for all 4 frames is then $(2^{22}/2^{54})^4 = 2^{-128}$. For 8 redundant bits per frame this is 2^{-32}.

This implies that using a relatively simple plaintext recognizer in the actual search units like in the DES cracker machines can produce good results if a similar design is used (the search units perform only simple plaintext recognition tests and the controlling PC then performs more complicated plaintext recognition tests for candidate keys).

5 Known Plaintext Attacks

Known plaintext attacks determine from a set of frames of ciphertext and the corresponding plaintexts a set of output bit streams (bit strings $y(t)|_{t=101}^{288}$) for A5/1, and from these attempt to compute the state of A5/1 before the mixing of the A5/1 frame counter.

The attacks all progress in the three steps below, for the random subgraph and biased birthday attacks the initial step is performed using a time/memory tradeoff algorithm.

1. Determine from the set of output bit streams the internal state of A5/1 for some cycle $t > 100$.
2. Reverse A5/1 to determine a set of possible initial states.
3. Reverse the mixing of the frame counter to determine a state of A5/1 with only the session key mixed in.

The state obtained in the last step can be used to generate a bit stream for any desired frame counter and therefore is sufficient to perform decryption and encryption for the session key.

5.1 Divide and Conquer Attack

A divide-and-conquer attack against A5/1 cipher is presented in [3]. The attack is a known plaintext attack and the objective of the attack is to recover the initial state $S(101)$ of A5/1 corresponding to a 64-bit prefix $y(101)...y(164)$ of known A5/1 output.

1. Generate 10 bits $x_{i,l}$, $0 \leq l < 10$, for each register i. Let $s_{i,\tau_i+l}(101) = x_{i,l}$.
2. For each register i based on the generated bits infer 10 linear equations. For all $\tau + l \leq r_i$ the equation is of the form $s_{i,\tau+l}(101) = x_{i,l}$ as above, otherwise infer the equation from the connection polynomial.

   For example $x_{1,10} = (s_{1,1}(101) + s_{1,2}(101) + s_{1,3}(101) + s_{1,6}(101)) \; mod \; 2$.

   The 30 equations are clearly linearly independent as an equation references one more register bit than the previous equation. Solving these equations will therefore solve 30 bits of the initial state.
3. The bits in the registers define now a linear equation involving the first plaintext bit $y(101)$ and the bits $s_{i,0}$.

   For example $s_{1,\tau_1}(101) = 1$, $s_{2,\tau_2}(101) = 1$ and $s_{3,\tau_3}(101) = 0$ implies $(s_{1,\tau_1}(101) + s_{2,\tau_2}(101)) \; mod \; 2 = y(101)$.
4. The bits $x_{i,l}$ define a set of state changes. The amount of state changes defined depends on the exact value of the bits due to the clock control. Each state change implies a linear equation involving a known plaintext bit and the output bits in the three registers. Each such equation will involve at least two new bits per state change and therefore the equations are linearly independent.

The equations are linearly independent from the equations in the previous step iff each equation contains a bit not present in any of the equations in the previous step. This is false only if R1 has been shifted in every state transition, which is highly improbable.

Assuming the bits are random, the probability that a register does not shift during a state change is $1/2 * 1/2 = 1/4$. Therefore the 10 bits in each register define an expected 40/3 state changes, generating an expected $40/3$ linear equations linearly independent with all the previous equations.

5. There are still approximately $63.32 - 13.30 - 1 \approx 19.02$ bits which values cannot unsolved for the initial state.

   Generate a tree of depth $19.02/3$. A node contains the next three input bits to the clock control function. A node has as children nodes that contain the next clock inputs that do not conflict with any of the linear equations already generated.

   The root has as children the nodes which contain the possible clock cycles not fully defined by the 30 bits generated in step 1.

   The expected number of children for each node will be 2.5 [3]. This means the amount of possible assignments to the bits not yet defined by the linear equations are $2.5^{4/3*19.02/3}$ with an expected value of approximately $2^{10.16}$.

The expected number of trials to find a correct state S(101) which generates the 64 bits of plaintext in question is now on average $2^{40.16}$. The algorithm should generate only few (presumably one) candidate states for S(101). [3]

5.2 Biased Birthday Attack

The biased birthday attack against A5/1 is a time/memory tradeoff and as such is divided into two phases. The first phase can be done independently of the session key used and the second phase is dependent of the session key. The objective of the attack is to determine the initial state of A5/1 for a cycle $101 \leq t \leq 277$

First compute a set A of A5/1 states which all produce as output a bit stream beginning with a common prefix $\alpha \in \{0,1\}^k$ for a fixed k. There are approximately 2^{64-k} such states. Call these states *special states*. This computation is described in the next subsection.

An output prefix of $k+\lceil log_2|A| \rceil$ bits are used to identify a special state with only relatively few expected collisions. Call an A5/1 initial state green if it generates the $k + \lceil log_2|A| \rceil$ bits identifying a special state in the output bit stream (bits $y(1)$ to $y(228)$). It is highly unlikely that α appears twice in the generated output bit stream and the map from green states to special states may be therefore considered to be many to one. Denote by the weight $W(s)$ of a special state s the amount of green states corresponding to the special state s.

[2] describes experiments performed regarding the weight of special states for $k = 16$. It shows $W(s)$ has a highly non-uniform distribution. For 85% of the special states $W(s) = 0$, for the rest of the states $1 \leq W(s) \leq 26000$. This implies that some special states will

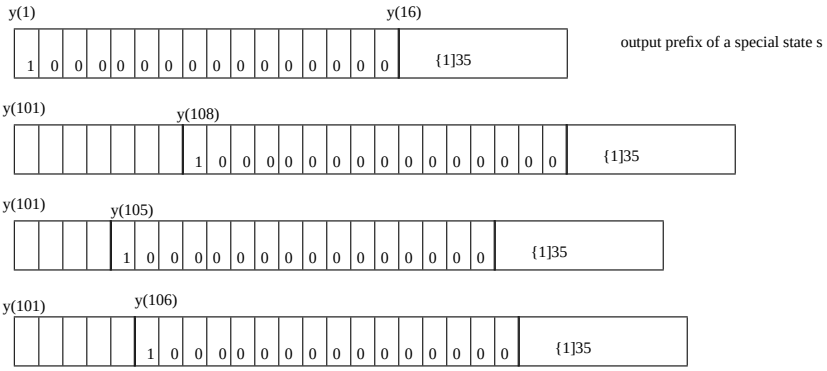


Figure 5: Example of a special state and three corresponding green states

be traversed with far higher probability than others during the generation of the 228 output bits.

There are approximately $100 + 114 + 114 - 100 - k - \lceil log_2|A| \rceil = 228 - k - \lceil log_2|A| \rceil$ offsets in the known output sequence that can contain the identifying prefix of a special state and hence there are approximately $2^{48} \times (201 - \lceil log_2|A| \rceil)$ green states.

Let $C \subseteq \{0,1\}^{228}$ be a set of known output sequences of A5/1 generated using the same session key. Let $B \subseteq C$ be the set of output sequences that contain the identifying prefix of a special state. $E[|B|] = 2^{-k} \times (228 - k - \lceil log_2|A| \rceil) \times |C|$ assuming that the set of output sequences for a session key are uniformly distributed over the set of frame counters. This approximation is justified in [2] by the mixing performed in the setup of the initial state.

Let $P_A(s)$ be the probability that special state $s \in A$ and $P_B(s)$ be the probability that there is an output sequence in B generated by a green state corresponding to s.

If A only contains special states s with weight $W(s) \geq w$ and $E[W(s)]$ is the expected weight of a special state in A then the expected number of special states occurring in both A and B is:

$$\Sigma_s P_A(s) \times P_B(s) = \Sigma_{s|W(s) \geq w} \frac{|B| \times W(s)}{(228 - k - \lceil log_2|A| \rceil) \times 2^{48}} = \frac{|A| \times |B| \times E[W(s)]}{(228 - k - \lceil log_2|A| \rceil \times 2^{48}}$$

[2] states that they were able to generate with $k = 16$ a set A such that $|A| = 2^{35}$ and $s \in A \Rightarrow E[W(s)] = 12500$. There are now $201 - 35 = 177$ offsets in the known output sequence that can contain the identifying prefix of a special state and therefore approximately $2^{48} \times 177$ green states.

The attack is based on colliding a special state in A to a green state corresponding to it in B. Assuming the cryptanalyst has access to a set C of known output sequences of A5/1 such that $|C| = 120 * 1000/4.6$ (approximately 2 min of GSM voice traffic), then $E[|B|] = 71$. The expected number of collisions is $\frac{E[|B|] \times 12500}{177 \times 2^{13}}$, for $E[|B|] = 71$ this is 0.61.

Upon detecting a collision is found, the cryptanalyst can determine the state or set states (in case the identifying output sequence was ambiguous) of A5/1 at a certain cycle t. The cryptanalysis now proceeds by reversing A5/1 to the initial state, reversing the mixing of

the frame counter and computing a state of A5/1 which contains only the mixed session key.

The test for a collision can be assumed to be performed in constant time, if the set A is ordered and indexed for efficient lookup. The attacker performs approximately $|B|$ lookups in A, with the success probability growing as $|B|$ grows. The attack can be performed in seconds on a PC. [2]

5.2.1 Pre-computation Phase

The purpose of the "pre-computation phase" is to produce a set of "special states" for a defined prefix α of length k. A special state produces as output a sequence beginning with the prefix α. There are approximately $2^{64} \times 2^{-k}$ special states for each prefix α. Call this process sampling A5/1 for special states.

The parameters used in the attack in [2] are $k = 16$.

1. Choose a prefix α of length k that does not coincide with shifted versions of itself, e.g. $\alpha = 100...0$.
2. Construct an arbitrary partial state of A5/1. The state is constructed by guessing the 19 bits of R1 and 11 bits of R2 and R3. The bits of R2 and R3 guessed are the clock control bits and the 10 next bits of R2 and R3 to enter the clock control next.
3. $19 + 11 + 11 = 41$ bits of the 64-bit state have now been defined. These bits are $s_{1,i}(0)$, $s_{2,11+j}(0)$ and $s_{3,12+j}(0)$ for $0 \leq i \leq 18$ and $0 \leq j \leq 10$. The prefix α defines $y(t)$ for $1 \leq t \leq k$.
4. For each state transition from a partial state there now exist at most 2 defined successor states. Let $t = 1$ and $(i, j, k) = (0, 0, 0)$.
   (a) If $s_{2,j}(0)$ and $s_{3,k}(0)$ are both defined and $s_{1,0}(t) \oplus s_{2,j}(0) \oplus s_{3,k}(0) \neq y(t)$ then the partial state can not be a special state. Discard the partial state.
   (b) If $s_{2,j}(0)$ is defined and $s_{3,k}(0)$ is not defined then let $s_{3,k}(0) = s_{1,0}(t) \oplus s_{2,j}(0) \oplus y(t)$.
   (c) If $s_{3,k}(0)$ is defined and $s_{2,j}(0)$ is not defined then let $s_{2,j}(0) = s_{1,0}(t) \oplus s_{3,k}(0) \oplus y(t)$.
   (d) If $s_{3,k}(0)$ and $s_{2,j}(0)$ are not defined, then there exist 2 valuations for $s_{2,j}(0)$ and $s_{3,k}(0)$ which satisfy $s_{3,k}(0) \oplus s_{2,j}(0) \oplus s_{1,0}(t) = y(t)$. Call the $s_{3,k}(0)$ and $s_{2,j}(0)$ "choice bits". Choose one of the valuations and define the choice bits.
   (e) Calculate the clock control (x, y, z) as per Table 2.
   (f) Let $i = i + x$, $j = j + y$, $k = k + z$ and $t = t + 1$. If $x = 1$ clock R1.
   (g) Iterate until $t > k$
5. If the state is not yet fully defined (there are still bits $s_{x,y}(0)$ that are undefined) then the undefined bits may be treated as choice bits and any assignment to them is valid.

Fraction	Avg Weight
2^{-4}	2432
2^{-5}	3624
2^{-6}	4719
2^{-7}	5813
2^{-8}	6910
2^{-9}	7991
2^{-10}	9181
2^{-11}	10277
2^{-12}	11369
2^{-13}	12456

Table 3: Average weights for the heaviest trees of various fractions of R

6. Iterate over all possible partial states and combinations of assignments to choice bits for each partial states to produce the set of all special states.

The algorithm clearly defines only special states and creates a set of all special states for a prefix α. There are 11 and 12 undefined bits and as A5/1 on average clocks each register with probability 0.75 the algorithm should encounter only a negligible amount of contradictions.

The case $E[W(s)] = 12500$ and $|A| = 2^{35}$ in A was generated by iterating over 2^{48} special states and choosing the heaviest special states. This may be too time-consuming for some PC's and the attacker may wish to trade shorter setup time for lower coverage of green states.

Table 3 shows results from [2] regarding the average weight of certain fractions of the heaviest special states. If the attacker is willing to halve the average weight of the trees in his set, the attacker could iterate over the special states and then on average every 2^{-8} would be acceptable and the pre-processing time could be reduced to $2^{35} \times 2^8 = 2^{43}$. This reduces the preprocessing time by a factor of $2^5 = 32$ and doubles the amount of known bit stream frames required for a succesfull attack.

5.2.2 Caching Special States

Special states can be identified by approximately 48 bits. The 41 guessed bits and the seven first choices made during the sampling of the special state. This identification is not necessarily unambiguous. If an ambiguous partial description of a state is encountered, then all possible states fullfilling the partial description must be considered. This is not a problem, as the A5/1 can be reversed and the session key candidate can be efficiently verified with another known bit stream frame.

The amount 2^{48} of special states is too high to cache these states on the disks of a modern PC. The purpose is to cover as much as possible of the space of green states via the special states being traversed during the generation of $y(101)...y(277)$.

The previous section shows how to generate sets of special states with certain properties.

The set A must be stored on a hard disk in practice. As $|A| = 2^{35}$ each byte required per special state requires 32 gigabytes of hard disk space.

The attack requires looking up the state from a $(prefix, state)$ for a known 51-bit prefix. Sort the pairs according to prefix in increasing order, and instead of the prefix store the increment for the prefix in between states and only periodically store the full 31 unknown bits of the prefix.

The amount of bits required to define a partial state can now be reduced as the 51-bit output prefix of the state is known. Reducing the amount of bits used to below 48 creates further possible ambiguity, but can be used to save hard disk space, although this is not necessary as $6 * 32 = 192GB$ of disk space can be acquired by using 3 75GB hard disks (e.g. IBM deskstar 75GXP)

A random disk access is estimated by [2] to require 6 ms. The biased birthday attack will make one such disk access for each 51-bit output prefix of a special state in C. If $|B| = 71$ then the time spent accessing disk is approximately 0.4 seconds.

5.3 Random Subgraph Attack

The random subgraph attack is another time/memory tradeoff attack against A5/1. The attack relies on the special states defined above, but attempts to cover all special states in the pre-computation phase, and hence requiring fewer frames of known A5/1 output. The objective of the attack is to compute a set of possible internal states of A5/1 corresponding to the state of A5/1 at different points in the known output sequence.

The random sub-graph attack is based on the Hellman time-memory tradeoff [2]. Let P be the set of plaintexts, K the set of keys and C the set of ciphertexts. Let $E_k : P \rightarrow C$ be the encryption function for key $k \in K$. Let $K = P = C = U$.

The idea of the attack is to define a function $f(K) = E_K(P)$ and a set of t permutations $\pi_i : U \rightarrow U$. Define $f_i(r) = \pi_i(f(r))$. Compute for m random values $r_n \in U$ the pairs $(r_n, f_i^t(r_n))$ for all functions f_i. See table 4 for an example.

Each pair $r_n, f_i^t(r_n))$ defines a sequence that covers 2^{12} of elements in U. Choosing a sufficient amount of such sequences from random *start points* to *end points* allows one to cover the set U with high probability.

The attack consists of iterating the functions f_i over the known plaintext $p \in U$ and for each iteration comparing the result with the end points in the computed table. If a collision is found, then the attack proceeds by iterating f_i from the start point corresponding to the determined end point until a collision is found with p. The key is with high likelihood the element in U used as input to the last iteration of f_i.

The random subgraph attack against A5/1 [2] is performed over the set of special sets R ($|R| = 2^{48}$). This is possible as each special set can be identified by the 41 bits and 7 choice bits used in the biased birthday attack.

Define $f : \{0,1\}^{48} \rightarrow \{0,1\}^{48}$. Let a be the special state identified by the 48-bit input. Initialize the internal state of A5/1 to this and clock A5/1 for 64 cycles. Now $y(1)...y(16) = \alpha$. Let bits $y(17)...y(64)$ be the result of $f(a)$. Define a set of 2^{12} random

	f_1	...	$f_{2^{12}}$
r_1	$(r_1, f_1^{2^{12}}(r_1))$	...	$(r_1, f_{2^{12}}^{2^{12}}(r_1))$
r_2	$(r_2, f_1^{2^{12}}(r_2))$	...	$(r_2, f_{2^{12}}^{2^{12}}(r_2))$
...			
$r_{2^{24}}$	$(r_{2^{24}}, f_1^{2^{12}}(r_{2^{24}}))$	...	$(r_{2^{24}}, f_{2^{12}}^{2^{12}}(r_{2^{24}}))$

Table 4: Pre-computed table for random sub-graph attack

permutations as r_i. Generate 2^{24} random values, for each random value generate 2^{12} (start point, end point) pairs as shown in table 4.

According to [2] a special state is contained in one of the generated sequences with high likelihood. The attack is performed against a frame containing the output sequence of any special state (identified by the 16-bit sequence α). The 48-bit sequence following α in the known output is then used as input to the 2^{12} different versions of A5/1 each of which is iterated for 2^{12} times over the input. For each iteration, the attacker tests for a collision with the set of computed end-points, and if a collision is found for a function f_i^k, the internal state of A5/1 is the preceding state $r_i^{-1}(f_i^{k-1}(\text{start point})$, which is straightforward to compute.

Table 4 is intended to be stored on a hard disk and [2] estimate the speed of a collision test to be 6ms. This would make the duration of an attack several days. A solution to this problem is proposed in [2]. Call a special state a *bright a red* state if the first 12 bits following α are all 0.

Generate the table now by iterating the functions f_i from the start point onwards until a bright red state is encountered (on average every 2^{12} special state is also a bright red state).

A bright red state can be generated by sampling special states and filtering out non bright red states, on average every 2^{12} state should be bright red. Now during the actual attack, one needs access the disk on average only for every 2^{12} special state, which makes the attack feasible. The time required for disk probes is now estimated at 24 seconds. [2]

After recovering an internal state of A5/1 and the cycle it is in, A5/1 can be reversed to recover a functional session key as in the previous attacks.

The attack requires a frame of known A5/1 output containing a 64-bit sequence beginning with the prefix α. The probability of a frame containing such a sequence is $(228 - 64) \times 2^{-16} = 164 \times 2^{-16}$. This means in practice 4.6 ms $\times 2^{16}/164 < 2$ s of GSM voice traffic.

The time complexity of the attack is 2^{24} assuming table lookups are performed in constant time. The attack can be performed on a PC in 4 minutes according to [2].

5.4 Reversing A5/1

After a set of initial states $S(t)$ for $101 \leq t \leq 327$ has been obtained the attack proceeds to determining the actual pre-mixing phase state S(0).

Each register $R1$, $R2$ and $R3$ has been clocked between 0 and $t - 1$ times during the mixing. Iterate through the $10^6 < 2^{20}$ possible combinations and see which ones create

initial states $S(0)$ that generate the known bit stream. The complexity of this attack is negligible compared to the previous attacks.

The problem of using special states with great weight might seem to imply that several candidate green states would be generated, but as the possible green states may be assumed to be uniformly distributed over the states $S(101), ..., S(277)$, even if a special state has weight 12500 then an expected amount of $12500/177 \approx 70$ candidate initial states for a fixed cycle t will be generated. [2]

Assuming that the contents of the clock control bits have been completely random, the estimated amount of clocks for each register is $0.75 * 101 = 76$. The attack may be speeded-up by attempting the most probable clock amounts first and the least probable last. The software implementation described earlier can be used to perform quick forwarding and reversing of A5/1. The expected amount of iterations for a candidate initial state are 10^4. [3]

The final step of the attack is to compute the session key. It is sufficient to reverse the mixing of the frame counter and compute S(-22). This state of A5/1 with only K_c mixed in is sufficient to encrypt and decrypt for K_c.

The expected amount of predecessors for a A5/1 state is 1 and varies between 0 and 4. The number of states leading to $S(0)$ therefore varies between 0 and 4^{22}. The expected size of predecessors grows linearly, and therefore the expected complexity of this attack is negligible compared to the earlier attacks.

Experiment results shown in [2] claim that they were unable to generate states for which there were more than 120 predecessors at depth 100. This supports the proposition that the frame counter mixing can be efficiently reversed, and even the actual session key can be recovered, if need be.

To determine if the computed state $S(-22)$ has been generated with the actual session key, mix $S(-22)$ with another frame counter and generate the $y(101)...y(328)$ output bits and compare to the corresponding bits for another frame. It is highly unlikely an incorrect candidate session key would generate equal bit streams with correct session key for two frames.

6 Conclusion

This paper surveyed the A5/1 cipher and several attacks proposed against it. All the considered attacks can be used to relatively easily claim that A5/1 is not a secure cipher in practice.

A brute-force attack against the cipher is deemed feasible using dedicated hardware similar to that used in the EFF DES Cracker machine, but would otherwise be impractical.

The divide and conquer attack by Golic requires solving approximately 2^{40} linear equations per session key and access to the 64 bits first bits of generated key stream for a single frame. No numbers were provided by Golic regarding it's implementation in practice and it is difficult to estimate it's actual speed.

The biased birthday attack by Biryukov, Shamir and Wagner is a time/memory tradeoff. The implementation can be tuned further to trade off pre-computation time and memory against required amounts of known key stream. The pre-computation requires 2^{40} to 2^{48} steps and 150GB to 300GB disk space. The actual attack requires in practice only seconds of execution time on a PC, but requires key stream frames for two or more minutes of GSM voice data.

The random subgraph attack by Biryukov, Shamir and Wagner is a time/memory tradeoff based on the Hellman time/memory tradeoff. The attack requires approximately 160 GB of disk space, and 4 minutes of execution time on a PC. The setup time consists of 2^{48} steps. The required amount of known bit stream frames corresponds to under 2 seconds of GSM voice data.

The two attacks above both require a significant effort during the setup, but can later be used to perform real-time attacks using a single PC, which makes the attacks especially devastating as the actual cryptanalytic attacks become cheap both in terms of time and money. This may facilitate making criminal eavesdropping on GSM conversations profitable.

References

[1] Marc Briceno, Ian Goldberg and David Wagner. A pedagogical implementation of A5, 10.05.1999 [referred 07.10.2000]
<http://www.scard.org/gsm/a51.html>

[2] Alex Biryukov, Adi Shamir and David Wagner. Real Time Cryptanalysis of A5/1 on a PC. 27.04.2000 [referred 07.10.2000]
<http://cryptome.org/a51-bsw.htm>

[3] Jovan Golic. Cryptanalysis of Alleged A5 Stream Cipher. In *Proc. of EUROCRYPT'97*, LNCS 1233, 239–255, Springer-Verlag

[4] Racal Research Ltd. Extracts from Technical Information GSM System Security Study. 10.6.1988 [referred 28.10.2000].
< http://jya.com/gsm061088.htm, Racal Research Ltd, 1988-06-10 >

[5] ETSI. GSM 03.20 v7.2.0 Digital Cellular Telecommunications System (Phase 2+): Security Related network functions. December 1999.

[6] ETSI. GSM 05.03 v6.1.3 Digital Cellular Telecommunications System (Phase 2+): Channel Coding. Mars 1999.

[7] EFF. Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design O'Reilly and Associates, 1998.