Another Attack on A5/1

Patrik Ekdahl and Thomas Johansson

***Abstract*— A5/1 is a stream cipher used in the GSM standard. Several time-memory trade-off attacks against A5/1 have been proposed, most notably the recent attack by Biryukov, Shamir and Wagner, which can break A5/1 in seconds using huge pre-computation time and memory. This paper presents a completely different attack on A5/1, based on ideas from correlation attacks. Whereas time-memory trade-off attacks have a complexity which is exponential with the shift register length, the complexity of the proposed attack is almost independent of the shift register length. Our implementation of the suggested attack breaks A5/1 in a few minutes using 2-5 minutes of conversation plaintext.**

***Index Terms*— A5/1, GSM, correlation attacks, cryptanalysis, stream ciphers.**

I. Introduction

A5/1 is the strong version of the encryption algorithm used in the GSM standard to provide privacy for more than 130 million customers in the air link of their voice and data communication. A sketch of the design of A5/1 was leaked in 1994, and the exact design was reversed engineered in 1999 by Briceno [3] from an actual GSM telephone. A5/1 is a stream cipher based on irregular clocking of three linear feedback shift registers. The key size is 64 bits and the keystream is produced by xoring the output from the three registers.

The first attacks were either "Guess-and-Determine" type of attacks, or time-memory tradeoff attacks, requiring about 2^{40}–2^{45} steps of computation [4]. At Fast Software Encryption 2000, Biryukov, Shamir and Wagner [2] presented an improved attack on A5/1, which required only a few seconds to a few minutes computation on a PC. This attack is based on the time-memory tradeoff attack by Golic [4], and requires storage of about 150-300 Gbyte of precomputed data as well as a long precomputation phase. Although this is a very impressive attack, it is still only a time-memory trade-off attack with expensive asymptotic behavior. If one would improve A5/1 by simply increasing the lengths of the shift registers, say twice as long, this kind of attack would no longer be possible to apply in practice. Another attack was presented at Indocrypt 2000 by Biham and Dunkelman [1]. There attack breaks the cipher within $2^{39.91}$ A5/1 clocking assuming $2^{20.8}$ bits of keystream available, but also this attack has an expensive asymptotic behavior. Recently, Krause [6] presented a general attack on LFSR-based stream ciphers (having certain properties), called the BDD-based cryptanalysis. This attack requires computational complexity of $n^{O(1)}2^{an}, a < 1$ polynomial time operations, where a is a constant depending on the cipher and n is the combined shift registers length. For A5/1, the attack archives $a = 0.6403$, so the complexity is again exponential in the shift registers length.

This paper presents a completely different attack on A5/1, based on ideas from correlation attacks, see e.g., [7], [8], [5]. It exploits a bad key initialization in A5/1, the fact that the key and the frame counter are initialized in a linear fashion. This "bad property" enables us to launch a type of correlation attack, which is quite powerful.

In opposite to all other attacks, this attack is (almost) independent of the shift register lengths. Instead, it depends on the number of times that the cipher is clocked before it starts producing the output bits. In A5/1 this number is 100. If this number is increased, the attack becomes weaker, and vice versa. The resulting attack breaks A5/1 in a few minutes without requiring any notable precomputation and without requiring huge storage.

The paper is organized as follows. In Section II a brief description of A5/1 is given, and in Section III we present the model and assumptions of the attack. In Section IV we present a basic correlation attack on A5/1. Section V refines the ideas and gives the full attack. In Section VI we present the simulation results for the attack, and finally, in Section VII we give some conclusions.

II. A Brief Description of A5/1

A GSM conversation is sent as a sequence of frames, where one frame is sent every 4.6 millisecond. Each frame contains 114 bits representing the communication from A to B, and another 114 bits representing the communication from B to A. Each conversation is encrypted by a new *session key* K. For each frame to be sent, the session key K is mixed with a publicly known *frame counter*, denoted F_n, and the result serves as the initial state of the shift registers in the A5/1 generator. It then produces 228 bits of running key, which is xored with the 228 bits of plaintext to produce the ciphertext.

A5/1 consists of three short binary linear feedback shift registers (LFSRs) of lengths 19, 22, 23, denoted by R1, R2, R3, respectively. The three LFSRs all have primitive feedback polynomials. The running key of A5/1 is given as the XOR of the output of the three LFSRs, as illustrated in Figure 1.

The LFSRs are clocked in an irregular fashion. It is a type of stop/go clocking with a majority rule as follows. Each register has a certain clocking tap, denoted $C1$, $C2$, $C3$, respectively. Each time the LFSRs are clocked, the three clocking taps $C1$, $C2$, $C3$ determine which of the LFSRs that are clocked. R1 and R2 are clocked, but not R3, if $C1 = C2 = C3+1$; R1 and R3 are clocked, but not R2, if $C1 = C2+1 = C3$; R2 and R3 are clocked, but not R1, if $C1+1 = C2 = C3$; finally, R1, R2 and R3 are all clocked, if $C1 = C2 = C3$. Note that at each step at least two LFSRs are clocked, and that the probability for an individual LFSR being clocked is $3/4$.

This work has been presented at the IEEE International Symposium on Information Theory (ISIT) 2001, Washington D.C., June 24 – 29, 2001.

The authors are with the Dept. of Information Technology, Lund University, P.O. Box 118, 221 00 Lund, Sweden, {patrik,thomas}@it.lth.se.

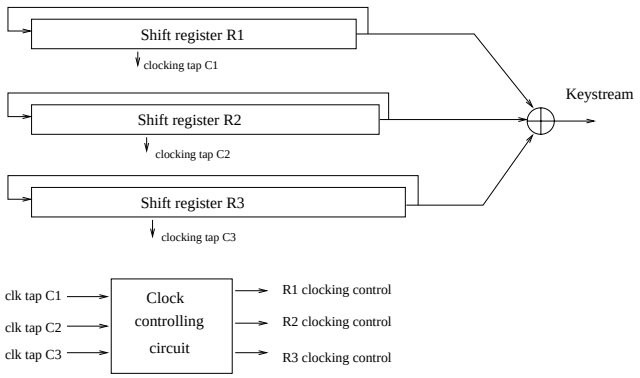

Fig. 1. Schematic description of A5/1

Finally, we describe the key initialization. First, the LFSRs are initialized to zero. They are then all clocked 64 times, ignoring the irregular clocking, and the key bits of K are consecutively xored in parallel to the feedback of each of the registers.

In the second step the LFSRs are clocked 22 times, ignoring the irregular clocking, and the successive bits from F_n are again xored in parallel to the feedback of each of the registers. Let us call the contents of the LFSRs at this time the *initial state* of the frame.

In the third step, the three registers are clocked for 100 additional clock cycles *with* the irregular clocking, but ignoring the output. Then finally, the three registers are clocked for 228 additional clock cycles with the irregular clocking, producing the 228 bits that form the running key.

For all other details around A5/1, like feedback polynomials etcetera, we refer to [2], where a more detailed description can be found.

III. The model for cryptanalysis

As mentioned in the previous section, the mode of operation for A5/1 is to xor the running key bits to the plaintext to produce the ciphertext. Denote the running key by $\mathbf{z} = z_1, z_2, \ldots$.

The standard model of an attack on a stream cipher is a *known-plaintext* attack. In this scenario, the attacker has access to N bits from both the ciphertext and the plaintext and thus, is able to deduce the N first bits of the running key $\mathbf{z}$.

In GSM, the digital representation of the conversation is split into frames of length 228 bits. The corresponding known-plaintext assumption is now that the attacker is given access to the running key from m different frames, each running key of length 228 bits. The attack presented here is based on these assumptions. Note that the frames are initialized with the same session key K but with different frame counters F_n.

Given the keystream, the goal of the proposed attack is to recover the initial state of the running key generator. This is called an *initial state recovery* attack. In the case of the GSM system the initial state of the shift registers is a linear combination of the publicly known frame counter and the secret session key. By deducing the initial state, we can recover the secret session key.

IV. A Basic Correlation Attack

Let us start by giving some fundamental observations. First, from the key initialization description we note that the initial state is a linear function of K and F_n. Let $K = (k_1, k_2, \ldots, k_{64})$ and $F_n = (f_1, f_2, \ldots, f_{22})$, where $k_i, f_i \in \mathbb{F}_2$. Let $u_0^1, u_1^1, \ldots$ be the LFSR output sequence produced by *regular* clocking of R1 after the key and frame number initialization (starting with the initial state). Similarly, let $u_0^2, u_1^2, \ldots$ be the LFSR output sequence produced by regular clocking of R2; and finally, let $u_0^3, u_1^3, \ldots$ be the LFSR output sequence of R3, when clocked regularly. This means that $(u_0^1, u_1^1, \ldots, u_{18}^1)$ forms the initial state of R1 for the given frame and similarly for R2 and R3.

Recall the linear fashion in which the key K and the frame number F_n together form the initial state of the frame. It is clear that the initial state of each LFSR can be expressed as a linear function of K and F_n. Given this observation, we can write each LFSR output symbol from R1 as

$$u_t^1 = \sum_{i=1}^{64} c_{it}^1 k_i + \sum_{i=1}^{22} d_{it}^1 f_i, \tag{1}$$

for some known binary constants c_{it}^1, $i = 1, \ldots, 64$, $t \geq 0$, and d_{it}^1, $i = 1, \ldots, 22$, $t \geq 0$. We introduce the notation $s_t^1 = \sum_{i=1}^{64} c_{it}^1 k_i$ and $\hat{f}_t^1 = \sum_{i=1}^{22} d_{it}^1 f_i$, $t \geq 0$. Then we can write

$$u_t^1 = s_t^1 + \hat{f}_t^1, \quad t \geq 0. \tag{2}$$

We call the sequence s_t^1 the *key part* of sequence u_t^1 and the sequence $\hat{f}_t^1$ is similarly called the frame number part of u_t^1. Of course, we can also write

$$s_t^1 = \sum_{i=0}^{18} \hat{c}_{it}^1 s_i^1, \tag{3}$$

for some known binary constants $\hat{c}_{it}^1$, $i = 0, \ldots, 18$, $t \geq 0$.

Note that $s_0^1, s_1^1, s_2^1, \ldots$ is an unknown binary sequence (2^{19} possible sequences) that remains the same for all frames within a conversation (it depends only on K, which is fixed during a conversation). Furthermore, $\hat{f}_1^1, \hat{f}_2^1, \ldots$, is a known contribution from the frame counter that is different for each frame. Since the frame number is always known, the above sequence $\hat{f}_1^1, \hat{f}_2^1, \ldots$, can be calculated for each frame. For registers R2 and R3 we can, in a similar way, write the output symbols as

$$u_t^2 = \sum_{i=1}^{64} c_{it}^2 k_i + \sum_{i=1}^{22} d_{it}^2 f_i, \tag{4}$$

$$u_t^3 = \sum_{i=1}^{64} c_{it}^3 k_i + \sum_{i=1}^{22} d_{it}^3 f_i, \tag{5}$$

with known binary constants c_{it}^2, c_{it}^3, $i = 1, \ldots, 64$,$t \geq 0$ and d_{it}^2, d_{it}^3, $i = 1, \ldots, 22$, $t \geq 0$. Following the notation for R1,

we introduce

$$s_t^2 = \sum_{i=1}^{64} c_{it}^2 k_i, \quad \text{and} \quad \hat{f}_t^2 = \sum_{i=1}^{22} d_{it}^2 f_i, t \geq 0,$$
$$s_t^3 = \sum_{i=1}^{64} c_{it}^3 k_i, \quad \text{and} \quad \hat{f}_t^3 = \sum_{i=1}^{22} d_{it}^3 f_i, t \geq 0,$$

for the key part and frame number part of the output sequences u_t^2 and u_t^3 of registers R2 and R3. Similarly to (2) we also write

$$u_t^2 = s_t^2 + \hat{f}_t^2, \quad t \geq 0, \tag{6}$$
$$u_t^3 = s_t^3 + \hat{f}_t^3, \quad t \geq 0. \tag{7}$$

Let us now present the very basic idea for a correlation attack on A5/1. It will later be considerably refined. Let $z_1, z_2, \ldots, z_{228}$ denote the observed running key from A5/1 in a certain frame. Let us consider what is happening after the LFSRs have received their initial values. First, the registers will be clocked irregularly for 100 times, producing no output, then they will be clocked once and the first output symbol will be produced. Since each of the shift registers will clock on average three times out of four, we can expect that after these 101 irregular clockings, each LFSR will have been clocked about 76 times. Assume for a moment that each of the three LFSRs has been clocked *exactly* 76 times. Then the produced bit z_1 is the XOR of the output of the three LFSRs,

$$u_{76}^1 + u_{76}^2 + u_{76}^3 = z_1. \tag{8}$$

Since $\hat{f}_1, \hat{f}_2, \ldots$ is a known quantity in each frame, we can simply calculate its contribution to the output bit. From (2), (6) and (7) we can rewrite (8) as

$$s_{76}^1 + s_{76}^2 + s_{76}^3 = \hat{f}_{76}^1 + \hat{f}_{76}^2 + \hat{f}_{76}^3 + z_1. \tag{9}$$

Note that the right hand side contains known quantities only. Denote the right hand side of (9) in frame j with $O^j_{(76,76,76,1)}$. Under the assumption that each LFSR has been clocked exactly 76 times, we get one bit of information about the key in frame j, since

$$s_{76}^1 + s_{76}^2 + s_{76}^3 = O^j_{(76,76,76,1)}. \tag{10}$$

Of course, if the assumption is not correct we can expect (10) to hold with probability $1/2$. Hence, we have identified a correlation by calculating

$$P(s_{76}^1 + s_{76}^2 + s_{76}^3 = O^j_{(76,76,76,1)}) = P(\text{assumption correct}) \cdot 1 + P(\text{assumption wrong}) \cdot 1/2. \tag{11}$$

The probability of all three LFSRs being clocked exactly 76 times in this case is calculated to be about 10^{-3}. Hence

$$P(s_{76}^1 + s_{76}^2 + s_{76}^3 = O^j_{(76,76,76,1)}) = 1/2 + 1/2 \cdot 10^{-3}. \tag{12}$$

The left-hand-side expression $s_{76}^1 + s_{76}^2 + s_{76}^3$ remains constant over all frames. It is not hard to show that if we have access to a few million frames, and thus can calculate $O^j_{(76,76,76,1)}$ for each frame, then $s_{76}^1 + s_{76}^2 + s_{76}^3$ can be determined with high confidence.

The value of $s_{76}^1 + s_{76}^2 + s_{76}^3$ will give us one bit of information about the key. By considering other assumed triples for the number of clockings of the three LFSRs, we can derive more information about the key and eventually recover it.

V. A Refinement of the Attack

The previously described attack is very simple, but has the drawback that it requires many frames. In this section we show how to considerably refine the attack.

Denote the produced keystream after the initialization with the key K and the frame counter F_n by $w_0, w_1, \ldots$. Recall that the first 101 symbols, $w_0, w_1, \ldots, w_{100}$ are discarded during the initialization and the running key z_i is given by $z_1 = w_{101}, z_2 = w_{102}, \ldots, z_{228} = w_{328}$. Consider a certain assumed clocking triple (cl_1, cl_2, cl_3) of registers R1, R2 and R3. This clocking might occur in several keystream positions, e.g. the clocking (79,79,79) might not only end up at position 101 but also at positions $102, 103, \ldots$, etc. Keystream positions before 101 are not considered since they are discarded and are not accessible. Let

$$P((cl_1, cl_2, cl_3) \text{ in } v\text{th position}) \tag{13}$$

denote the probability of clocking (cl_1, cl_2, cl_3) occurring at position v (i.e., keystream symbol w_v).

Given a specific clocking triple (cl_1, cl_2, cl_3), we can, calculate an interval $\mathcal{I}$ for v, where that clocking triple has a non-negligible probability of occurring. So, instead of using only *one* keystream position when calculating the correlation probability as done in (11) and (12), we can use *all* positions ($v \geq 101$) where there is a non-negligible probability of occurrence. In frame j we calculate a correlation probability (implicitly conditioned on $\mathcal{I}$ in the jth frame), denoted $p^j_{(cl_1,cl_2,cl_3)} = P(s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3} = 0)$, as a weighted voting over several positions using the formula:

$$p^j_{(cl_1,cl_2,cl_3)} = \sum_{v \in \mathcal{I}} P((cl_1, cl_2, cl_3) \text{ in } v\text{th position}) \cdot [O^j_{(cl_1,cl_2,cl_3,v-100)} = 0] + 1/2 \cdot (1 - \sum_{v \in \mathcal{I}} P((cl_1, cl_2, cl_3) \text{ in } v\text{th position})), \tag{14}$$

where $[x = 0]$ is the indicator function which equals 1 if $x = 0$ and 0 otherwise. Also, $O^j_{(cl_1,cl_2,cl_3,v-100)} = \hat{f}^1_{cl_1} + \hat{f}^2_{cl_2} + \hat{f}^3_{cl_3} + z_{v-100}$ as in (9)-(10), for frame j. Finally, $\mathcal{I}$ is the interval where there is a non-negligible probability of occurrence for the clocking triple (cl_1, cl_2, cl_3).

If we assume that the bits entering the clock controlling device of A5/1 are uniformly distributed independent bits, we can write the probability (13) as a recursive formula,

$$P((cl_1, cl_2, cl_3) \text{ in } v\text{th position}) = F(cl_1, cl_2, cl_3, v), \tag{15}$$

TABLE I
COMPARISON OF THE APPROXIMATED VALUE GIVEN BY (15) OR (16) AND THE ESTIMATED VALUE FROM SIMULATIONS OF THE A5/1 CIPHER. ALL VALUES SHOULD BE MULTIPLIED BY 10^{-4}.

$P \cdot 10^{-4}$	Eq. (15) or Eq. (16)	Estimation by simulation
P(76,76,76,101)	9.7434	9.7331
P(79,79,79,105)	9.2012	9.2033
P(80,80,80,105)	6.6388	6.6388
P(79,80,81,106)	8.3858	8.4126
P(82,82,82,109)	8.7076	8.7269

where

$$
\begin{aligned}
&F(cl_1, cl_2, cl_3, 0) = 1 \text{ if } cl_1 = 0, cl_2 = 0 \text{ and } cl_3 = 0,\\
&F(cl_1, cl_2, cl_3, v) = 0 \text{ if } cl_1 < 0 \text{ or } cl_2 < 0 \text{ or } cl_3 < 0,\\
&F(cl_1, cl_2, cl_3, v) = 0 \text{ if } cl_1 > v \text{ or } cl_2 > v \text{ or } cl_3 > v,\\
&F(cl_1, cl_2, cl_3, v) =\\
&\qquad 0.25F(cl_1 - 1, cl_2 - 1, cl_3 - 1, v - 1)\\
&\qquad + 0.25F(cl_1, cl_2 - 1, cl_3 - 1, v - 1)\\
&\qquad + 0.25F(cl_1 - 1, cl_2, cl_3 - 1, v - 1)\\
&\qquad + 0.25F(cl_1 - 1, cl_2 - 1, cl_3, v - 1).
\end{aligned}
$$

This formula will give an exact probability under the assumption of independent uniformly distributed clocking bits. We will use these probabilities to approximate the actual A5/1 case. The approximation works well when the probability is fairly high (as in the considered cases in this paper), since we then have several different initial states that give the desired (cl_1, cl_2, cl_3, v).

Under the same assumptions as above, we can give a somewhat easier formula which gives a closed expression for the probability in (13) as

$$
P((cl_1, cl_2, cl_3) \text{ in } v\text{th position}) = \frac{\binom{v}{v-cl_1}\binom{v-(v-cl_1)}{v-cl_2}\binom{v-(v-cl_1)-(v-cl_2)}{v-cl_3}}{4^v}. \quad (16)
$$

The expression in (16) gives the same value as (15) for any *valid* clocking triple (cl_1, cl_2, cl_3) in position v, e.g. it will fail for $(0,0,0)$ in position 10 which cannot happen in A5/1. Table I gives an indication of the validness of the approximation compared to an estimated value based on 100 million simulation of the A5/1 cipher.

We give a small fictitious example to clarify (14) for three different sequences of values $O^j_{(cl_1,cl_2,cl_3,v-100)}, v = 101, 102, ..., 106$.

Example 1: As presented in Figure 2, we have chosen $\mathcal{I} = \{101, \ldots, 106\}$. We see from the tabulated example that if we calculate the sequence $O^j_{(cl_1,cl_2,cl_3,v-100)}, v = 101, 102, ..., 106$, using the known frame number and the running key $\mathbf{z}$, to be only zeros in the interval $\mathcal{I}$, then the probability that the key part $s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3}$ for this specific clocking is zero is fairly high (0.9). If we calculate the same sequence to be only ones, the probability of the key part being zero is low (0.1). Finally, if we have a mix of ones and zeros we see that the zeros are observed at positions where there is a (in total) higher probability of occurrence, so in this case we vote for $s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3}$ being zero, due to a slightly higher probability (0.62).

Keystream pos. v

	100	101	102	103	104	105	106	107
	..	z_1	z_2	z_3	z_4	z_5	z_6	...
$P((cl_1, cl_2, cl_3)$ in vth position) =		0.04	0.16	0.20	0.20	0.16	0.04	
$O^j =$		0	0	0	0	0	0	$p^j_{(cl_1,cl_2,cl_3)} = 0.9$
$O^j =$		1	1	1	1	1	1	$p^j_{(cl_1,cl_2,cl_3)} = 0.1$
$O^j =$		1	0	1	0	0	1	$p^j_{(cl_1,cl_2,cl_3)} = 0.62$

Fig. 2. Example of three different sequences $O^j_{(cl_1,cl_2,cl_3,v)}$ and the corresponding $p^j_{(cl_1,cl_2,cl_3)}$ probabilities calculated according to (14).

In order to use the information in all the available frames to estimate the value of the linear combination $s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3}$ we will introduce a log-likelihood ratio. First, define $\hat{p}_{(cl_1,cl_2,cl_3)} = P(s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3} = 0)$ as the total probability that $s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3} = 0$, taken over all frames. Recall that $p^j_{(cl_1,cl_2,cl_3)}$ denoted the same for the jth frame only. Then define the log-likelihood ratio $\Lambda_{(cl_1,cl_2,cl_3)}$ of $\hat{p}_{(cl_1,cl_2,cl_3)}$ as

$$
\Lambda_{(cl_1,cl_2,cl_3)} = \ln \frac{\hat{p}_{(cl_1,cl_2,cl_3)}}{1 - \hat{p}_{(cl_1,cl_2,cl_3)}}, \quad (17)
$$

where ln is the natural logarithm. We can now calculate an estimate of $\Lambda_{(cl_1,cl_2,cl_3)}$ over all frames by calculating

$$
\Lambda_{(cl_1,cl_2,cl_3)} = \sum_{j=1}^{m} \ln \frac{p^j_{(cl_1,cl_2,cl_3)}}{1 - p^j_{(cl_1,cl_2,cl_3)}}, \quad (18)
$$

where m is the number of available frames. For a log-likelihood ratio Λ defined as in (17) we know that $\Lambda = 0$ if $P(s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3} = 0) = 1/2$, and $\Lambda > 0$ if $P(s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3} = 0) > 1/2$, and finally $\Lambda < 0$ if $P(s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3} = 0) < 1/2$.

We will now turn to specific parameter choices as we describe the final phase of the attack. Starting at position 79, we pick a suitable interval of length 8, $\mathcal{C}_1 = \{79, \ldots, 86\}$, and look at all linear combinations of $s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3}$ where each of cl_1, cl_2, cl_3 runs in the interval $\mathcal{C}_1$. For each such value of (cl_1, cl_2, cl_3) and for each frame $j = 1, \ldots, m$, we calculate $p^j_{(cl_1,cl_2,cl_3)}$ and use (18) to calculate $\Lambda_{(cl_1,cl_2,cl_3)}$. Using $\Lambda_{(cl_1,cl_2,cl_3)}$ we finally estimate the linear combination of key bits with a simple hard decision. For example, if $\Lambda_{(79,79,79)} = 2.56$ we estimate $s^1_{79} + s^2_{79} + s^3_{79} = 0$, if $\Lambda_{(79,79,80)} = -0.93$ then $s^1_{79} + s^2_{79} + s^3_{80} = 1$, etc.

We note that when (cl_1, cl_2, cl_3) runs through all possible values in the specified interval of size 8, this gives a system of $8^3 = 512$ linear equations with $8 + 8 + 8 = 24$ unknown variables. The problem of finding the correct values of the 24 unknown variables is now equivalent to the problem of decoding a length 512 linear code of dimension 24. The

estimated bits can be viewed as a received word of length 512, and the corresponding equations are parity check equations for the code. If we have enough frames to make the estimates reasonably accurate, we can decode the received word and find 24 bits from the key.

Since the log-likelihood ratio Λ represents a soft value of the probability, it is also possible to use a soft decoding algorithm. A soft decoding algorithm would be expected to perform better since it takes more advantage of the given information. We have tried soft decoding but it did not improve the attack notably. When the number of received frames increases, the probability tends to either 0 or 1 quite fast and thus reducing the gain of soft decoding. The reduced complexity of the hard decoding algorithm seems to be a better choice in this case.

Table II shows the average, maximum and minimum number of correct estimates of the 512 equations, in a run of 60 simulations, using the procedure described.

TABLE II
NUMBER OF CORRECT ESTIMATES FOR A SYSTEM OF 512 EQUATIONS.

	Number of thousand frames in the estimation (m).					
	10	30	50	70	100	200
Average:	283	293	309	320	326	354
Max:	308	307	330	347	346	378
Min:	259	283	288	303	301	339

Using the interval $\mathcal{C}_1 = \{79, \ldots, 86\}$ we solve for the bits $s^1_{79}, \ldots, s^1_{86}$, $s^2_{79}, \ldots, s^2_{86}$ and $s^3_{79}, \ldots, s^3_{86}$ from the key part of the registers. Using (3) this will give us $8 + 8 + 8 = 24$ bits of information about the key K (in the form of linear combinations of key bits). To fully recover the key (64 bits) we can increase the length of the interval to 22, such that we get $22 + 22 + 22 > 64$ bits of information about the key, which makes the decoding much harder. Instead, we propose to pick a new subinterval $\mathcal{C}_2 = \{87, \ldots, 94\}$, thus recovering another 24 bits from the key. Finally, we do the same for the subinterval $\mathcal{C}_3 = \{95, \ldots, 102\}$. Then we have recovered 24 bits from each shift register output and a total of 72 bits. This is more than what we need for solving for the key K.

The computational work to check a solution consists of first loading the estimated bits into the register, then running the cipher backwards 79 clocks plus additional 22 clocks for the frame number loading. Then loading the frame number in the usual way and run the 100 premix clocks and finally checking the generated keystream output against the received keystream. The maximum number of output bits that we need to check is about 64, since the state space is 64 bits. The computational work of checking a solution thus sums to one register loading plus about $223 + 64 = 287$ cipher clockings.

VI. SIMULATIONS OF THE ATTACK

In order to check the correctness of the attack proposed in the previous section, we have implemented it. In a first step, the probabilities $p^j_{(cl_1,cl_2,cl_3)}$ are calculated for each frame $j = 1, \ldots, m$ and cl_1, cl_2, cl_3 each in the interval $\mathcal{I}$, and the $\Lambda_{(cl_1,cl_2,cl_3)}$ log-likelihood ratios are calculated.

In a second step, the decoding in our simulations is done by exhaustive search over all possible values of $s^1_{i_1}, s^2_{i_2}, s^3_{i_3}$, where i_1, i_2, i_3 each runs through the interval $\mathcal{I}$. The solution which gives the closest Hamming distance to the received codeword is taken as the correct solution. However, in order to have a high probability that the correct solution is the codeword closest in Hamming distance to the received word, we still need quite many frames. Simulations show that when we have fewer than about 100 000 frames there are often other (erroneous) solutions to the system of equations that give a closer distance. To overcome this problem we save a list of the $T \approx 1000$ closest solutions for each subinterval. Picking one solution from each list(subinterval), we can combine them into three 24 bit LFSR sequences as allegedly produced by the shift registers. These sequences are then verified by running the cipher backwards as described above.

In the case of using an interval length of 8 we need three subintervals and the number of combinations to verify would amount to T^3, which is rather expensive. A more efficient way is to use overlapping intervals where each subinterval overlaps the previous subinterval with 2 or 3 bits. Now we only have to verify combinations that agree in the overlapping bits. We can also use the fact that the sequences we want to verify are 24 bits long and the shift registers are shorter. Thus, a first test if the combined sequence could be the right, is to check whether the last bits of the sequence fulfill the feedback polynomial of each of the shift registers.

Using these techniques, simulations with $T = 1000$ have shown that we can reduce the number of verifications from 1000^3 to about $5000 - 50000$ for the case when the interval size is 8 and number of overlapping bits is 3. The different configurations we used in our simulations are shown in Table III.

TABLE III
CONFIGURATIONS USED IN OUR SIMULATIONS.

Interval size	Overlapping bits	Intervals
7	3	$[79, 85]$, $[83, 89]$, $[87, 93]$, $[91, 97]$, $[95, 101]$
8	3	$[79, 86]$, $[84, 91]$, $[89, 96]$, $[94, 101]$
9	2	$[79, 87]$, $[86, 94]$, $[93, 101]$

Table IV shows the success rate for different configurations and different number of received frames. The entries are the number of successful attacks out of a batch of 100 runs and in parenthesis are the attack times for each configuration. The corresponding length of the GSM conversation is also given (although this is a known plaintext attack and would not apply directly to the GSM system). The simulations where run on a PC with Intel Pentium 4 processor, running at 1.8Ghz, with 512 MByte of memory using a Linux operating system.

The precomputation phase in the presented attack amounts to calculate and store the probabilities that a certain number of clockings of the registers appears in a certain keystream position. These are the probabilities used in (14). Using the same hardware as in our simulations, it takes about 15 minutes to calculate the required tables and less that 2Mb to

TABLE IV
SIMULATION RESULTS USING A LIST SIZE OF $T = 1000$. ENTRIES SHOW THE NUMBER OF SUCCESSES OUT OF 100 RUNS. TIME OF ATTACK IS GIVEN IN PARENTHESIS.

# Success / (time of attack)	Number of received frames (time of GSM conversation in min/sec)		
Config.	30000 (2m30s)	50000 (3m45s)	70000 (5m20s)
7/3	2/(1min)	13/(2min)	49/(3min)
8/3	2/(2min)	20/(3min)	57/(4min)
9/2	3/(3min)	33/(4min)	76/(5min)

store them. We have summarized the implemented attack in Figure 3.

1) Pick a subinterval $\mathcal{C}_1$ (e.g., $\mathcal{C}_1 = [79, 86]$)
2) Let (cl_1, cl_2, cl_3) run through the interval $\mathcal{C}_1$. For each frame $j = 1, \ldots, m$ calculate

$$p^j_{(cl_1,cl_2,cl_3)} = \sum_{v=\mathcal{I}} P((cl_1, cl_2, cl_3) \text{ in } v\text{:th pos.}) \cdot [O^j_{(cl_1,cl_2,cl_3,v-100)} = 0] + 1/2 \cdot (1 - \sum_{v=\mathcal{I}} P((cl_1, cl_2, cl_3) \text{ in } v\text{:th pos.}))$$

Calculate the log-likelihood ratio of the weighted probability over all frames

$$\Lambda_{(cl_1,cl_2,cl_3)} = \sum_{i=1}^{m} \ln \frac{p^j_{(cl_1,cl_2,cl_3)}}{1 - p^j_{(cl_1,cl_2,cl_3)}}$$

Estimate the linear combination

$$s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3} = \text{HD}(\Lambda_{(cl_1,cl_2,cl_3)})$$

using a hard decision (HD) on the value of $\Lambda_{(cl_1,cl_2,cl_3)}$.
3) Decode the generated linear code

$$s^1_{cl_1} + s^2_{cl_2} + s^3_{cl_3} = \text{HD}(\Lambda_{(cl_1,cl_2,cl_3)})$$

for (cl_1, cl_2, cl_3) in interval $\mathcal{C}_1$ using a ML decoding through exhaustive search. Save the T closest solutions.
4) Repeat steps 1 to 3 for each new subinterval $\mathcal{C}_2, \mathcal{C}_3, \ldots$ until totally 64 bits of the shift register sequences are recovered.
5) Combine the solutions from each subinterval and check the validness of the solutions.

Fig. 3. A summary of the proposed attack.

Recalling (14), the summation is taken over an interval $\mathcal{I}$ where there is a non-negligible probability of occurrence of clocking (cl_1, cl_2, cl_3). Calculating the probabilities for the highest clocking, $(101, 101, 101)$, in the simulations, shows that this clocking has a very small probability of occurring beyond the $v = 140$th keystream position (40th position in the running key). So the attack only needs the first 40 bits of the keystream in each frame. Furthermore, the frames used in the attack need not to be consecutive.

VII. CONCLUSIONS

We have proposed a new attack on the A5/1 stream cipher, based on an identified correlation. In contrast to previous attacks, this is not a time-memory trade-off attack, but uses completely different properties of the cipher. It explores the weak key initialization which allows to separate the session key from the frame number in binary linear expressions.

The complexity of the attack is only linear in the length of the shift registers and depends instead on the number of irregular clockings before the keystream is produced. The implemented attack needs the 40 first bits from about 2^{16} (possible non-consecutive) frames, which corresponds to about 5min of GSM conversation. Our implementation of the attack shows that we have a high success rate; more than 70%. This can be improved by using larger list size and/or larger interval size. The complexity of the attack using the parameters presented here is quite low and the attack can be carried out on a modern PC in less than 5 minutes using very little precomputation time and memory.

The improvements compared to previous work are the following. All previous attacks have a complexity exponential in the shift register length. The complexity of the attack presented in this paper is roughly linear in the shift register lengths.

Previous attacks also need either much precomputation and/or memory or they have a high time complexity. The proposed attack is simple to implement, has been implemented, and completes its task in less than 5 minutes.

Finally, the presented attack also enlightens new interesting design weaknesses in A5/1 that should be considered when constructing new stream ciphers.

REFERENCES

[1] E. Biham, O. Dunkelman, "Cryptanalysis of the A5/1 GSM stream Cipher", *Lecure Notes in Computer Science*, vol. 1977, 2000, pp. 43–51, (Indocrypt 2000).
[2] A. Biryukov, A. Shamir, D. Wagner, "Real time cryptanalysis of A5/1 on a PC", *Lecture Notes in Computer Science*, vol. 1978, 2001, pp. 1–18, (FSE'2000).
[3] M. Briceno, I. Goldberg, D. Wagner, "A pedagogical implementation of A5/1", `http://scard.org`, May 1999.
[4] J. Golic, ''Cryptanalysis of alleged A5 stream cipher", *Lecture Notes in Computer Science*, vol. 1233, 1997, pp. 239–255, (Eurocrypt'97).
[5] T. Johansson, F. Jönsson, "Improved fast correlation attacks on stream ciphers via convolutional codes", *Lecture Notes in Computer Science*, vol. 1592, 1999, pp. 347-362, (Eurocrypt'99).
[6] M. Krause "BDD-based Cryptanalysis of Keystream Generators", to be presented at EUROCRYPT 2002, available at IACR eprint server *http://www.iacr.org*.
[7] W. Meier, and O. Staffelbach, "Fast correlation attacks on certain stream ciphers", *Journal of Cryptology*, vol. 1, 1989, pp. 159–176.
[8] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography* , CRC Press, 1997.

VIII. BIOGRAPHIES

Patrik Ekdahl (S'98) was born in Malmö, Sweden, on April 3, 1972. He received his M.Sc. in Electrical Engineering from Lund University, Lund, Sweden in 1998. In May 1998 he became a graduate student at the Department of Information

Technology at Lund University, where he is currently working on his Ph.D. thesis.

His research interests are mainly in cryptology, in particular analysis and design of stream ciphers.

Thomas Johansson (S'92-M'95) was born in Ljungby, Sweden, in 1967. He received the M.Sc. degree in computer science in 1990 and the Ph.D. degree in information theory in 1994, both from Lund University, Lund, Sweden.

Since 1995, he has held various teaching and research positions at the Department of Information Technology at Lund University. Since 2000, he has been Professor of Information Theory at the same department. His scientific interests include cryptology, error-correcting codes, and information theory.

Prof. Johansson has served on cryptologic program committees such as EUROCRYPT'98 '00 '01' 02, FSE'01 '02, etc. He was a recipient of the SSF-JIG (Junior Individual Grant).