Conditional Estimators: An Effective Attack on A5/1

Elad Barkan and Eli Biham

Computer Science Department,
Technion – Israel Institute of Technology,
Haifa 32000, Israel
{barkan, biham}@cs.technion.ac.il
http://www.technion.ac.il/∼barkan/
http://www.cs.technion.ac.il/∼biham/

Abstract. Irregularly-clocked linear feedback shift registers (LFSRs) are commonly used in stream ciphers. We propose to harness the power of conditional estimators for correlation attacks on these ciphers. Conditional estimators compensate for some of the obfuscating effects of the irregular clocking, resulting in a correlation with a considerably higher bias. On GSM's cipher A5/1, a factor two is gained in the correlation bias compared to previous correlation attacks. We mount an attack on A5/1 using conditional estimators and using three weaknesses that we observe in one of A5/1's LFSRs (known as $R2$). The weaknesses imply a new criterion that should be taken into account by cipher designers. Given 1500–2000 known-frames (about 4.9–9.2 conversation seconds of known keystream), our attack completes within a few tens of seconds to a few minutes on a PC, with a success rate of about 91%. To complete our attack, we present a source of known-keystream in GSM that can provide the keystream for our attack given 3–4 minutes of GSM ciphertext, transforming our attack to a ciphertext-only attack.

1 Introduction

Correlation attacks are one of the prominent generic attacks on stream ciphers. There were many improvements to correlation attacks after they were introduced by Siegenthaler [13] in 1985. Many of them focus on stream ciphers composed of one or more regularly clocked linear feedback shift registers (LFSRs) whose output is filtered through a non-linear function. In this paper, we discuss stream ciphers composed of irregularly-clocked linear feedback shift registers (LFSRs), and in particular, on stream ciphers whose LFSRs' clocking is controlled by the mutual value of the LFSRs. The irregular clocking of the LFSRs is intended to strengthen the encryption algorithm by hiding from the attacker whether a specific register advances or stands still. Thus, it should be difficult for an attacker to correlate the state of an LFSR at two different times (as he does not know how many times the LFSR has been clocked in between).

Assume the attacker knows the number of clocks that the LFSRs have been clocked until a specific output bit has been produced. The attacker can guess the

B. Preneel and S. Tavares (Eds.): SAC 2005, LNCS 3897, pp. 1–19, 2006.

number of clocks that the LFSRs are clocked during the generation of the next output bit with some success probability $p < 1$. A better analysis that increases the success probability of guessing the number of clocks for the next output bit could prove devastating to the security of the stream cipher. Our proposed conditional estimators are aimed at increasing this success probability.

In this paper, we introduce conditional estimators, aimed to increase the probability of guessing the clockings of the LFSRs correctly. We apply conditional estimators to one of the most fielded irregularly clocked stream ciphers — A5/1, which is used in the GSM cellular network. GSM is the most heavily deployed cellular phone technology in the world. Over a billion customers world-wide own a GSM mobile phone. The over-the-air privacy is currently protected by one of two ciphers: A5/1 — GSM's original cipher (which was export-restricted), or A5/2 which is a weakened cipher designated for non-OECD (Organization for Economic Co-operation and Development) countries. As A5/2 was discovered to be completely insecure [3], the non-OECD countries are now switching to A5/1. The internal design of A5/1 and A5/2 was kept secret until Briceno [6] reverse engineered their internal design in 1999. Contrary to A5/1 and A5/2, the internal design of the future GSM cipher A5/3 was officially published.

The first attacks on A5/1 were proposed by Golic [9] in 1997, when only a rough design of the cipher was leaked. He proposed two known-keystream attacks: the first is a guess and determine attack, and the second is a time-memory tradeoff attack. In 2000, the second attack was significantly improved by Biryukov, Shamir, and Wagner [5]. In some scenarios the improved attack can find the key in less than a second. However, the attack requires four 74-gigabyte disks and a lengthy precomputation. At the same time, Biham and Dunkelman [4] took a different approach. Their attack requires a few tens of seconds of known-keystream and recovers the key with a time complexity of about 2^{40} A5/1 cycles. In 2003, Barkan, Biham, and Keller [3] showed a ciphertext-only attack that finds the encryption key of A5/2, using the fact that in GSM the error-correction codes are employed before encryption. They converted this attack to an active attack on A5/1 networks, and also presented a ciphertext-only time-memory tradeoff attack on A5/1. However, the latter requires a very lengthy precomputation step. As for correlation attacks, in 2001, Ekdahl and Johansson [7] applied ideas from correlation attacks to A5/1. Their attack requires a few minutes of known-keystream, and finds the key within minutes on a personal computer. In 2004, Maximov, Johansson, and Babbage [11] discovered a correlation between the internal state and the output bits and used it to improved the attack. Given about 2000–5000 frames (about 9.2–23 seconds of known-plaintext), their attack recovers the key within 0.5–10 minutes on a personal computer.

These attacks demonstrate that fielded GSM systems do not provide an adequate level of privacy for their customers. However, breaking into fielded A5/1 GSM systems using these attacks requires either active attacks (e.g., man in the middle), a lengthy (although doable) precomputation step, a high time complexity, or a large amount of known keystream.

One advantage of correlation attacks on A5/1 over previous attacks is that they require no long-term storage and no preprocessing, yet given a few seconds of known-keystream, they can find the key within minutes on a personal computer. Another advantage of correlation attacks over some of the previous attacks is the immunity to transmission errors. Some of the previous attacks are susceptible to transmission errors, e.g., a single flipped bit defeats Golic's first attack. Correlation attacks can naturally withstand transmission errors, and even a high bit-error-rate can be accommodated for.

In this paper, we introduce conditional estimators, which can compensate for some of the obfuscating effects caused by the irregular clocking. Using conditional estimators, we improve the bias of the correlation equation that was observed in [11] by a factor of two. In addition, we discover three weaknesses in one of A5/1's registers. We mount a new attack on A5/1 based on the conditional estimators and the three weaknesses. Finally, we describe a source for known keystream transforming our attack to a ciphertext-only attack.

One of the weaknesses relates to the fact that register $R2$ of A5/1 has only two feedback taps, which are adjacent. This weakness enables us to make an optimal use of the estimators by translating the problem of recovery of the internal state of the register to a problem in graph theory. Thus, unlike previous attacks [7, 11], which were forced to use heuristics, we can exactly calculate the list of most probable internal states. We note that in 1988, Meier and Staffelbach [12] warned against the use of LFSRs with few feedback taps. However, it seems that their methods are difficult to apply to A5/1.

An alternative version of our attack can take some advantage of the fact that many operators set the first bits of the key to zero (as reported in [6]); this alternative version slightly simplifies the last step of our attack, and results with a somewhat higher success rate. We are not aware of any other attack on A5/1 (except for exhaustive search) that could benefit from these ten zero bits.

Our last contribution is a new source for known-plaintext in GSM. We point at the Slow Associated Control CHannel (SACCH) and show that its content can be derived. We also discuss the frequency hopping in GSM and how to overcome it. Using this new source for known-plaintext, our attacks can be converted to ciphertext-only attacks. However, this is a slow channel, that provides only about eight known frames each second.

We have performed simulations of our attacks. Given 2000 frames, our simulations take between a few tens of seconds and a few minutes on a PC to find the key with a success rate about 91%. For comparison, the simulations of [11] with a similar number of frames take about four times longer to run and achieve a lower success rate of about only 5%. A comparison of some of the results of previous works and our results is given in Table 1. With our new source for known keystream, the required 1500–2000 *known frames* can be obtained from the *ciphertext* of about 3–4 minutes of conversation.

This paper is organized as follows: We give a short description of A5/1 in Section 2, then, we set our notations and review some of the main ideas of previous works in Section 3. In Section 4 we describe the conditional estimators

Table 1. Comparison Between the Our Attacks and Previous Works. Only passive attacks are included, i.e., the active attack of [3] is not shown. The attack time for [3, 4, 5] is our estimate. As [3, 5] are time/memory/data tradeoff attacks, we give the tradeoff point that uses data that is equivalent to four minutes of ciphertext.
* based on error-correction codes as described in [3] (not on Section 7).
♯ preprocessing time

Attack: (Configuration explained in Section 6)	Required Frames: Known Keystream	Required Frames: Ciphertext Only (by Section 7)	Average Time on a single PC (range)	Success Rate
Ekdahl & Johansson [7]	70000 (322 s)	140 min	5 min	76%
[7]	50000 (230 s)	99 min	4 min	33%
[7]	30000 (138 s)	60 min	3 min	3%
Biham & Dunkelman [4]	20500 (95 s)	40.8 min	≈ 1.5 days	63%
Maximov et al. [11]	10000 (46 s)	20 min	10 min	99.99%
[11]	10000 (46 s)	20 min	76 s	93%
[11]	5000 (23 s)	10 min	10 min	85%
[11]	5000 (23 s)	10 min	44 s	15%
[11]	2000 (9.2 s)	4 min	10 min	5%
[11]	2000 (9.2 s)	4 min	29 s	1%
Biryukov et al. [5]	2000 (9.2 s)	4 min	♯ > 5 years	
Ciphertext only attack of Barkan et al. [3]	—	4 min*	♯ > 2300 years	
This Paper	2000 (9.2 s)	4 min	(6–10 min)	64%
early filtering (220000, 40000, 2000, 5200)	2000 (9.2 s)	4 min	(55–300 s)	64%
early filtering (100000, 15000, 200, 300)	2000 (9.2 s)	4 min	(32–45 s)	48%
improved estimators, (200000, 17000, 900, 2000)	2000 (9.2 s)	4 min	74 s (50–145 s)	86%
improved estimators, (200000, 36000, 1400, 11000)	**2000 (9.2 s)**	**4 min**	**133 s (55–626s)**	**91%**
early filtering (120000, 35000, 1000, 800)	1500 (6.9 s)	3 min	(39–78 s)	23%
improved estimators, (88000, 52000, 700, 1200)	1500 (6.9 s)	3 min	82 s (44–105 s)	48%
improved estimators, (88000, 52000, 3200, 15000)	**1500 (6.9 s)**	**3 min**	**7.2 min (44–780 s)**	**54%**

and three weaknesses, and then use them in our new attack in Section 5. The results of our simulations are presented in Section 6. We describe the new source of known-plaintext in Section 7. Finally, the paper is summarized in Section 8.

2 A Description of A5/1

The stream cipher A5/1 accepts a 64-bit session key K_c and a 22-bit publicly-known frame number f. GSM communication is performed in frames, where a frame is transmitted every 4.6 millisecond. In every frame, A5/1 is initialized with the session key and the frame number. The resulting 228 bit output (keystream) is divided into two halves: the first half is used to encrypt the data from the network to the mobile phone, while the second half is used to encrypt the data from the mobile phone to the network. The encryption is performed by XORing the data with the appropriate half of the keystream.

A5/1 has a 64-bit internal state, composed of three maximal-length Linear Feedback Shift Registers (LFSRs): $R1$, $R2$, and $R3$, with linear feedbacks as shown in Figure 1. Before a register is clocked the feedback is calculated (as the XOR of the feedback taps). Then, the register is shifted one bit to the right (discarding the rightmost bit), and the feedback is stored into the leftmost location (location zero).

A5/1 is initialized with K_c and f in three steps, as described in Figure 2. This initialization is referred to as the *key setup*.

Observe that the key setup is linear in the bits of both K_c and f, i.e., once the key setup is completed, every bit of the internal state is an XOR of bits in fixed locations of K_c and f. This observation is very helpful in correlation attacks.

A5/1 works in cycles, where in each cycle one output bit is produced. A cycle is composed of irregularly clocking $R1$, $R2$, and $R3$ according to a clocking mechanism that we describe later, and then outputting the XOR of the rightmost bits of the three registers (as shown in Figure 1). The first 100 bits of output are discarded, i.e., the 228 bits that are used in GSM are output bits 100,...,327. The keystream generation can be summarized as follows:

1. Run the key setup with K_c and f (Figure 2).
2. Run A5/1 for 100 cycles and discard the output.
3. Run A5/1 for 228 cycles and use the output as keystream.

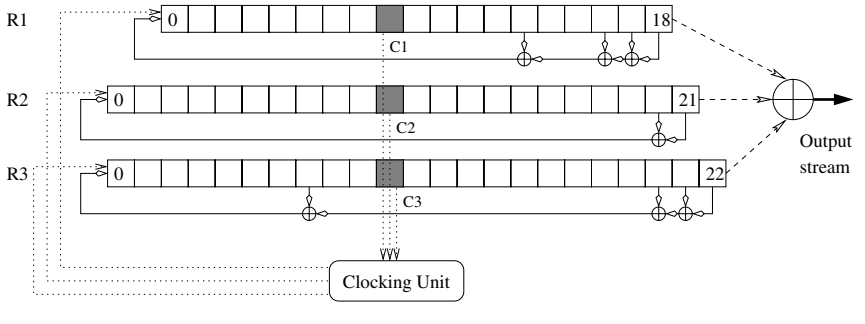

Fig. 1. The A5/1 internal structure

1. Set $R1 = R2 = R3 = 0$.
2. For $i = 0$ to 63
   - Clock all three registers.
   - $R1[0] \leftarrow R1[0] \oplus K_c[i]$; $R2[0] \leftarrow R2[0] \oplus K_c[i]$; $R3[0] \leftarrow R3[0] \oplus K_c[i]$.
3. For $i = 0$ to 21
   - Clock all three registers.
   - $R1[0] \leftarrow R1[0] \oplus f[i]$; $R2[0] \leftarrow R2[0] \oplus f[i]$; $R3[0] \leftarrow R3[0] \oplus f[i]$.

Fig. 2. The key setup of A5/1. The i'th bit of K_c is denoted by $K_c[i]$, and the i'th bit of f is denoted by $f[i]$, where $i = 0$ is the least significant bit. We denote the internal state after the key setup by $(R1, R2, R3) = \text{keysetup}(K_c, f)$.

It remains to describe the clock control mechanism, which is responsible for the irregular clocking. Each register has a special *clocking tap* near its middle (in locations $R1[8]$, $R2[10]$, and $R3[10]$). The clocking mechanism algorithm:

1. Calculate the majority of the values in the three clocking taps.
2. Then, clock a register if and only if its clocking tap agrees with the majority.

For example, assume that $R1[8] = R2[10] = c$ and $R3 = 1-c$ for some $c \in \{0, 1\}$. Clearly, the value of the majority is c. Therefore, $R1$ and $R2$ are clocked, and $R3$ stands still.

Note that in each cycle of A5/1, either two or three registers are clocked (since at least two bits agree with the majority). Assuming that the clocking taps are uniformly distributed, each register has a probability of 1/4 for standing still and a probability of 3/4 for being clocked.

3 Notations and Previous Works

In this section, we set our notations, and describe some of the main ideas of the previous works. Let S_1, S_2, and S_3 be the initial internal state of registers $R1$, $R2$, and $R3$ after the key-setup (using the correct K_c), where the frame number is chosen to be zero, i.e., $(S_1, S_2, S_3) = \text{keysetup}(K_c, 0)$. For $i = 1, 2, 3$, denote by $\tilde{S}_i[l_i]$ the output bit of Ri after it is clocked l_i times from its initial state S_i.[1] Similarly, let F_1^j, F_2^j, and F_3^j be the initial internal state of registers $R1$, $R2$, and $R3$ after a key setup using all zeros as the key, but with frame number j, i.e., $(F_1^j, F_2^j, F_3^j) = \text{keysetup}(0, j)$. For $i = 1, 2, 3$, denote by $\tilde{F}_i^j[l_i]$ the output of Ri after it is clocked l_i times from its initial state F_i^j. Ekdahl and Johansson [7] observed that due to the linearity of the key setup, the initial internal value of Ri at frame j is given by $S_i \oplus F_i^j$, i.e., $\text{keysetup}(K_c, j) = \text{keysetup}(K_c, 0) \oplus \text{keysetup}(0, j) = (S_1 \oplus F_1^j, S_2 \oplus F_2^j, S_3 \oplus F_3^j)$. Furthermore,

[1] Note that as a register has a probability of 3/4 of being clocked in each cycle, it takes about $l_i + l_i/3$ cycles to clock the register l_i times.

due to the linear feedback of the shift register, the output of LFSR i at frame j after being clocked l_i times from its initial state is given by $\tilde{S}_i[l_i] \oplus \tilde{F}_i^j[l_i]$.

Maximov, Johansson, and Babbage [11] made the following assumptions:

1. *clocking assumption* (j, l_1, l_2, t): Given the keystream of frame j, registers $R1$ and $R2$ were clocked exactly l_1 and l_2 times, respectively, until the end of cycle t. The probability that this assumption holds is denoted by $Pr((l_1, l_2)$ at time $t)$ (this probability can be easily computed, see [11]).
2. *step assumption* (j, t): Given the keystream of frame j, both $R1$ and $R2$ are clocked in cycle $t+1$, but $R3$ stands still. Assuming the values in the clocking taps are uniformly distributed, this assumption holds with probability 1/4 (the clocking mechanism ensures that if the values of the clocking taps are uniformly distributed, each register stands still with probability 1/4).

They observed that under these two assumptions, $R3$'s contribution to the output is fixed in output bits t and $t+1$. Thus, $R3$ does not affect the difference between these two output bits, and the following equation holds:

$$(\tilde{S}_1[l_1] \oplus \tilde{S}_2[l_2]) \oplus (\tilde{S}_1[l_1+1] \oplus \tilde{S}_2[l_2+1]) = \\ \tilde{Z}^j[t] \oplus \tilde{Z}^j[t+1] \oplus (\tilde{F}_1^j[l_1] \oplus \tilde{F}_2^j[l_2]) \oplus (\tilde{F}_1^j[l_1+1] \oplus \tilde{F}_2^j[l_2+1]), \quad (1)$$

where $\tilde{Z}^j[t]$ is the output bit of the cipher at time t of frame j. Thus, the value of $(\tilde{S}_1[l_1] \oplus \tilde{S}_2[l_2]) \oplus (\tilde{S}_1[l_1+1] \oplus \tilde{S}_2[l_2+1])$ can be estimated from the known keystream and the publicly available frame numbers.

Equation (1) holds with probability 1 if both the clocking assumption and the step assumption hold. If either or both assumptions do not hold, then Equation (1) is assumed to hold with probability 1/2 (i.e., it holds by pure chance). Therefore, Equation (1) holds with probability $(1 - Pr((l_1, l_2)$ at time $t))/2 + Pr((l_1, l_2)$ at time $t)((3/4)/2 + 1/4) = 1/2 + Pr((l_1, l_2)$ at time $t)/8$. The bias $Pr((l_1, l_2)$ at time $t)/8$ is typically two to three times higher compared to the bias shown in [7]. Such a difference in the bias is expected to result in an improvement of the number of frames needed by a factor between four and ten, which is indeed the case in [11].

We simplify Equation (1) by introducing the notation $\tilde{S}'_i[l_i]$ defined as $\tilde{S}_i[l_i] \oplus \tilde{S}_i[l_i+1]$. Similarly denote $\tilde{F}_i^j[l_i] \oplus \tilde{F}_i^j[l_i+1]$ by $\tilde{F}_i'^j[l_i]$, and denote $\tilde{Z}^j[t] \oplus \tilde{Z}^j[t+1]$ by $\tilde{Z}'^j[t]$. Thus, Equation (1) can be written as:

$$(\tilde{S}'_1[l_1] \oplus \tilde{S}'_2[l_2]) = \tilde{Z}'^j[t] \oplus (\tilde{F}_1'^j[l_1] \oplus \tilde{F}_2'^j[l_2]) \quad (2)$$

Observe that due to the linearity of the LFSR, $\tilde{S}'_i[l_i]$ can be viewed as the output of Ri after it has been clocked l_i times from the initial state $S'_i \triangleq S_i^+ \oplus S_i$, where S_i^+ denotes the internal state of Ri after it has been clocked once from the internal state S_i. Note that there is a one-to-one correspondence between S_i and S'_i, therefore, when we recover S'_i, we find S_i.

In [11] it was observed that better results are obtained by working with d consecutive bits of the output of S'_i, where d is a small integer. A *symbol* is defined to be the binary string of d consecutive bits $S'_i[l_i] \triangleq \tilde{S}'_i[l_i]||\tilde{S}'_i[l_i+1]|| \cdots ||\tilde{S}'_i[l_i+d-1]$,

where "||" denotes concatenation. For example, $S'_2[81] = \tilde{S}'_2[81]$ is a 1-bit symbol, and $S'_1[90] = \tilde{S}'_1[90]||\tilde{S}'_1[91]$ is a 2-bit symbol.

In the first step of [11], estimators are calculated based on the above correlation and on the available keystream. For every pair of indices l_1 and l_2 for which estimators are computed, and for every possible symbol difference $\delta = S'_1[l_1] \oplus S'_2[l_2]$, the estimator $E_{l_1,l_2}[\delta]$ is the logarithm of the a-posteriori probability that $S'_1[l_1] \oplus S'_2[l_2] = \delta$. For example, for $d = 1$, the symbol is a single bit, thus, the symbol difference can be either zero or one. Then, for $l_1 = 80$ and $l_2 = 83$, the estimator $E_{80,83}[0]$ is the logarithm of the probability that $S'_1[80] \oplus S'_2[83] = 0$, and $E_{80,83}[1]$ is the logarithm of the probability that $S'_1[80] \oplus S'_2[83] = 1$. For $d = 2$, there are four estimator for every pair of indices, e.g., $E_{80,83}[00_2]$, $E_{80,83}[01_2]$, $E_{80,83}[10_2]$, and $E_{80,83}[11_2]$ (where "$_2$" denotes the fact that the number is written in its binary representation, e.g., 11_2 is the binary representation of the number 3). The value of $E_{80,83}[10_2]$ is the logarithm of the probability that $S'_1[80] \oplus S'_2[83] = 10_2$, and so on. Note that the higher d is — the better the estimators are expected to be (but the marginal benefit drops exponentially as d grows).

In order to save space, we do not describe here how to calculate the estimators given the known-keystream and d. See [11] for the details. We would only note that the time complexity of this step is proportional to 2^d. With 2000 frames, the simulation in [11] takes about eleven seconds to complete this step with $d = 1$, and about 40 seconds with $d = 4$.

The rest of the details of previous works deal with how to decode the estimators and to recover candidate values for S_1, S_2, and S_3 (and thus recovering the key). These methods are basically heuristic methods that decode the estimators in short intervals of l_i of the output of S'_i, and then intersect the resulting candidates to find candidates for S_1, S_2, and S_3.

4 The New Observations

In this section, we describe tools and observations that we later combine to form the new attack.

4.1 The New Correlation — Conditional Estimators

In Section 3, we reviewed the correlation equation used by Maximov, Johansson, and Babbage. This correlation equation is based on two assumptions, the clocking assumption and the step assumption. Recall that the step assumption (i.e., that the third register stands still) holds in a quarter of the cases (assuming that the values in the clocking taps are independent and uniformly distributed).

Consider registers $R1$ and $R2$, and assume that for a given frame j and output bit t the clocking assumption holds, i.e., we know that $R1$ and $R2$ were clocked l_1 and l_2 times, respectively, from their initial state. Also assume that we know the value of $\tilde{S}_1[l_1 + 10]$ and $\tilde{S}_2[l_2 + 11]$. We use the publicly known frame number j to find the value of the clocking taps $C_1 = \tilde{S}_1[l_1 + 10] \oplus \tilde{F}_1^j[l_1 + 10]$ of $R1$ and $C_2 = \tilde{S}_2[l_2 + 11] \oplus \tilde{F}_2^j[l_2 + 11]$ of $R2$ at output bit t.

We observe that the bias of the correlation can be improved by a factor of two by dividing the step assumption into two distinct cases. The first of the two distinct cases is when $C_1 \neq C_2$. Due to the clocking mechanism, $R3$ is always clocked in this case along with either $R1$ or $R2$. The step assumption does not hold, and therefore, Equation (2) is assumed to hold in half the cases. In other words, the case where $C_1 \neq C_2$ provides us no information.

However, in the second case, when $C_1 = C_2$, we gain a factor two increase in the bias. In this case, both $R1$ and $R2$ are clocked (as $c = C1 = C2$ is the majority), and $R3$ is clocked with probability 1/2, in case its clocking tap $C3 = c$ (we assume that the values of the clocking taps are uniformly distributed). Therefore, when $C_1 = C_2$, the step assumption holds with probability 1/2 compared to probability 1/4 in [11].

We analyze the probability that Equation (2) holds when $C1 = C2$. If either the step assumption or the clocking assumption do not hold, then we expect that Equation (2) holds with probability 1/2 (i.e., by pure chance). Together with the probability that the assumptions hold, Equation (2) is expected to hold with probability $Pr((l_1, l_2) \text{ at time } t)(1/2{+}1/2{\cdot}1/2){+}1/2(1{-}Pr((l_1, l_2) \text{ at time } t)) = 1/2 + Pr((l_1, l_2) \text{ at time } t)/4$ compared to $1/2 + Pr((l_1, l_2) \text{ at time } t)/8$ in [11]. Therefore, when $C_1 = C_2$, we gain a factor two increase in the bias compared to [11].[2]

We use the above observation to construct *conditional estimators* (which are similar to conditional probabilities). We define a d-bit *clock symbol* $S_i[l_i]$ in index l_i as the d-bit string: $\tilde{S}_i[l_i]||\tilde{S}_i[l_i{+}1]||\cdots||\tilde{S}_i[l_i{+}d{-}1]$, where "||" denotes concatenation. The conditional estimator $E_{l_1,l_2}[x|Sc]$ for indices l_1, l_2 is computed for every possible combination of a clock symbol difference $Sc = S_1[l_1{+}10] \oplus S_2[l_2{+}11]$ and a symbol difference $x = S'_1[l_1] \oplus S'_2[l_2]$. The estimator $E_{l_1,l_2}[x|Sc]$ is the logarithm of the a-posteriori probability that the value of the symbol difference is x, given that the value of the clock symbol difference is Sc. The computation of conditional estimators is similar to the computation of the estimators as described in [11], taking into account the above observations. The complete description of the calculation of conditional estimators will be given in the full version of this paper.

One way of using conditional estimators is to remove the conditional part of the estimators, and use them as regular estimators, i.e., compute $E_{l_1,l_2}[x] = \log\left(\frac{1}{2^d}\sum_y e^{E_{l_1,l_2}[x|y]}\right)$. Nevertheless, the benefit would not be large. A better use of the conditional estimators is to use them directly in the attack as is shown in Section 5.1.

4.2 First Weakness of $R2$ — The Alignment Property

The first weakness of $R2$ uses the fact that the feedback taps of $R2$ coincide with the bits that are estimated by the correlation equation. Assume that the

[2] As a refinement of these observations, note that it suffices to know the value of $\tilde{S}_1[l_1 + 10] \oplus \tilde{S}_2[l_2 + 11]$, since we only consider $C_1 \oplus C_2$ rather than the individual value of $C1$ and $C2$.

value of S_1 is known. Then for every index i, the correlation equation estimates the value of $S_2[i] \oplus S_2[i+1]$. On the other hand the linear feedback of $R2$ forces $S_2[i] \oplus S_2[i+1] = S_2[i+22]$. Thus, the correlation equation actually estimates bits which are 22 bits away. Using our notations, this property can be written as

$$S'_2[i] = S_2[i+22].$$

4.3 Second Weakness of $R2$ — The Folding Property

The second weakness of $R2$ is that it has only two feedback taps, and these taps are adjacent. Let $X[*]$ be a bit-string which is an output of $R2$, and let $cost(i, x)$ be a cost function that sets a cost for every possible d-bit string x in index i of the string $X[*]$ (the cost function is independent of the specific stream $X[*]$). We calculate the total cost of a given string $X[*]$ (i.e., calculate its "score") by

$$\sum_i cost(i, X[i]||X[i+1]||\cdots||X[i+d-1]). \tag{3}$$

Given the cost function, we can also ask what is the string X_{max} that maximizes the above sum, i.e., the string with the highest score.

The folding property allows to create a new cost function $cost'(i, x)$, where i is one of the first 22 indices. The special property of $cost'$ is that the score calculated on the first 22 indices using $cost'$ is equal to the score using Equation (3) over all the indices (using $cost$). $cost'$ is very helpful in finding the highest scored string X_{max} for a given cost function $cost$. However, the transition from $cost$ to $cost'$ has the penalty that $cost'(i, x)$ operates on d'-bit strings x that are slightly longer than d. In general, every 22 additional indices (beyond the first 22 indices) in $X[*]$ add one bit of length to x (in our simulation we work with strings of 66 indices, therefore, our $cost'$ operates on strings of length $d' = d + 2$).

For every index i, it holds that $X[i+22] = X[i] \oplus X[i+1]$, due to the linear feedback taps of $R2$. Therefore, the d'-bit string at index i determines a $(d'-1)$-bit string at index $i+22$, a $(d'-2)$-bit string at index $i+2\cdot 22$, a $(d'-3)$-bit string at index $i+3\cdot 22$, etc. Clearly, the contribution to the score of the strings in these indices is also determined by the value of the d'-bit string at index i, and thus can be "folded" into the cost function for index i.

For simplicity, we assume that the number of indices is divisible by 22, i.e., $22k + d - 1$ bits of $X[*]$ are included in the score computation (the attack can easily be extended to cases where the number of indices is not divisible by 22). The calculation of $cost'$ from $cost$ is given in Figure 3. We call the d'-bit strings *representative symbols*. Note that not every choice of 22 representative symbols is a consistent output of $R2$, as the 22 representative symbols span $22 + d' - 1$ bits (and thus there are $2^{22+d'-1}$ possibilities for these bits), while $R2$'s internal state has 22 bits. Specifically, the last $d' - 1$ bits are determined by the first d' bits through the linear feedback. Denote these last $d' - 1$ bits by w.

The linear feedback of $R2$ is actually calculating the difference between adjacent bits. We denote this operation using the difference operator D, i.e., $D(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{d'}) = (\alpha_1 \oplus \alpha_2, \alpha_2 \oplus \alpha_3, \ldots, \alpha_{d'-1} \oplus \alpha_{d'})$.

For each $i \in \{i_s, \ldots, i_s + 21\}$
  For each $e \in \{0,1\}^{d+k-1}$
    $cost'(i,e) \triangleq \sum_{j=0}^{k-1} cost(i + 22k, \mathrm{lsb}_d(D^j(e))$

Fig. 3. The folding property; calculating $cost'$ from $cost$. Denote the first index of $X[*]$ by i_s, and the number of indices by $22k$. $D(x)$ is the difference function that calculates the difference string — the parity of each two adjacent bits in x; $D^j(x)$ is j applications of D on x. $\mathrm{lsb}_d(x)$ returns the d least significant bits of x, thus, $\mathrm{lsb}_d(D^j(e))$ is the d-bit string in index $i + 22k$ that is determined by e.

For the first bits to be consistent with the last bits w, we require that the first bits are equal to $D_0^{-1}(w)$ or $D_1^{-1}(w)$, where $D_0^{-1}(w)$ is the value such that $D(D_0^{-1}(w)) = w$, and the first bit of $D_0^{-1}(w)$ is zero (i.e., D_0^{-1} is one of two inverses of D). $D_1^{-1}(w)$ is the 1-complement of $D_0^{-1}(w)$ (it also satisfies $D(D_1^{-1}(w)) = w$, i.e., D_1^{-1} is the other inverse of D).

4.4 Third Weakness of *R2* — The Symmetry Property

The third weakness in $R2$ is that its clock tap is exactly in its center. Combined with the folding property, a symmetry between the clocking tap and the output tap of $R2$ is formed. The symmetry property allows for an efficient attack using conditional estimators. Assume that S_1 is known. $S_2[i]$ is at the output tap of $R2$ when $S_2[i+11]$ is at the clock tap. When $S_2[i+11]$ reaches the output tap, $S_2[i+11+11] = S_2[i+22]$ is at the clock tap. However, the representative symbol at i determines both the bits of $S_2[i]$ and $S_2[i+22]$. Therefore, the representative symbols are divided into pairs, where each pair contains a representative symbol of some index i and a representative symbol of index $i+11$. When the representative symbols of index i serve for clocking, the other representative symbol is used for the output, and vice versa, i.e., the representative symbols in the pair control the clocking of each other. If the clocking taps were not in the middle, we could not divide the representative symbols into groups of two.

5 The New Attack

The attack is composed of three steps:

1. Compute the conditional estimators.
2. Decode the estimators to find list of best candidate pairs for S_1, S_2 values, by translating the problem of finding the best candidates to a problem in graph-theory.
3. For each candidate in the list, recover candidates for S_3. When a triplet S_1, S_2, S_3 is found, the key is recovered and verified through trial encryptions.

The computation of conditional estimators is based on Section 4.1, and similar to the computation of estimators in [11]. We will give a full description of this computation in the full version of the paper. Step 2 is described in Section 5.1.

In Step 3, given candidate pairs for S_1 and S_2, we work back candidates for S_3 from the keystream. The method is similar to the one briefly described by Ross Anderson in [1]. However, some adjustments are needed as the method in [1] requires the internal state right at the beginning of the keystream (after discarding 100 bits of output), whereas Step 2 provides candidates for the internal state after the key setup but before discarding 100 bits of output (the candidates for S_1 and S_2 XORed with F_1^j and F_2^j, respectively, are the internal state right after the key-setup and before discarding 100 bits of output). An alternative Step 3 exhaustively tries all 2^{23} candidate values for S_3. Taking into account that many operators set ten bits of the key to zero (as reported in [6]), we need to try only the 2^{13} candidate values for S_3 which are consistent with the ten zero bits of the key. A more detailed description of Step 3 will be given in the full version of this paper.

5.1 Step 2 — Decoding of Estimators

The aim of Step 2 is to find the list of best scored candidates for S_1 and S_2, based on the conditional estimators. The score of s_1 and s_2 (candidate values for S_1 and S_2, respectively) is simply the sum of their estimators (which is the logarithm of the product of the a-posteriori probabilities), i.e.,

$$score(s_1, s_2) = \sum_{l_1, l_2} E_{l_1,l_2}[s'_1[l_1] \oplus s'_2[l_2] \quad | \quad s_1[l_1 + 10] \oplus s_2[l_2 + 11]].$$

The list of best candidates is the list of candidates $\{(s_1, s_2)\}$ that receive the highest values in this score. For the case of non-conditional estimators, the *score* is defined in a similar manner but using non-conditional estimators (instead of conditional estimators).

Surprisingly, the list of best candidate pairs can be efficiently computed using the three weaknesses of $R2$. We translate the problem of calculating the list of best scored candidates into a problem in graph theory. The problem is modeled as a huge graph with a source node s and target node t, where each path in the graph from s to t corresponds to a candidate value for S_1 and S_2, with the score of the pair being the sum of the costs of the edges along the path (also, for every candidate pair s_1, s_2, there is a single path in the graph from s to t). Thus, the path with the heaviest score ("longest" path) corresponds to the highest scored pair. A Dijkstra-like algorithm [2] (for finding shortest path) can find the longest path, since the weights on the edges in our graph are negative (logarithm of probability). The list of best candidates corresponds to the list of paths which are near the heaviest. The literature for graph algorithms dealt with finding N-shortest paths in a graph (e.g., [10]); these algorithms can be adapted to our graph, and allow to find the heaviest paths.

Our graph contains 2^{19} subgraphs, one for each candidate value for S_1. All the subgraphs have the same structure, but the weights on the edges are different. Each such subgraph has one incoming edge entering the subgraph from the source node s, and one outgoing edge from the subgraph to the target node t. Both edges have a cost of zero.

5.1.1 The Structure of the Sub-graph Using Non-conditional Estimators

Our method for decoding the estimators can be used with non-conditional estimators, and in fact the structure of the subgraph is best understood by first describing the structure of the subgraph for the case of non-conditional estimators. In this case, the subgraph for the j^{th} candidate of S_1 has a source node s_j and a target node t_j. The subgraph is composed of $2^{d'-1}$ mini-subgraphs. Each mini-subgraph corresponds to one combination w of the last $d'-1$ bits of the representative symbol in index i_s+21 (last representative symbol). Figure 4 shows an example of a subgraph for $d'=3$, in which only the mini-subgraph for $w=01$ is shown. The full subgraph contains a total of four mini-subgraphs, which differ only in the locations of the two incoming edges (and their weight) and the outgoing edge. For each index $i \in \{i_s, \ldots, i_s+21\}$, the mini-subgraph includes $2^{d'-1}$ nodes: one node for each combination of last $d'-1$ bits of the representative symbols in index i. A single outgoing edge connects the mini-subgraph relevant node ${}^0_1 01$ in index i_s+21 to t_j (the other nodes in index i_s+21 can be erased from the mini-subgraph). Two incoming edges (for $D_0^{-1}(w)$ and $D_1^{-1}(w)$) connect s_j to relevant nodes in index i_s, which in our example are $D_0^{-1}(01)=001$ and $D_1^{-1}(01)=110$ (the nodes ${}^0_1 00$ and ${}^0_1 11$ in index i_s can be erased from the mini-subgraph). Thus, any path that goes through the mini-subgraph must include one of these incoming edges and the outgoing edge. This fact ensures that each path corresponds to a consistent choice of representative symbols (as discussed at the end of Section 4.3).

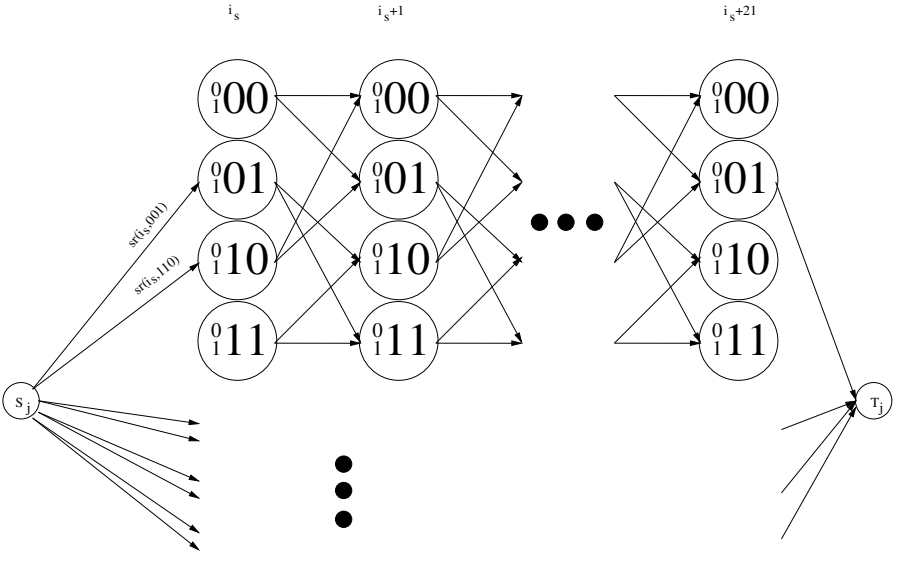

Fig. 4. The subgraph for the j^{th} candidate value of S_1

Consistent transitions between representative symbols in adjacent indices are modeled by edges that connect nodes of adjacent indices (in a way that reminds a de-Bruijn graph). There is an edge from a first node to a second node if and only if the last $d'-1$ bits of the first node are the same as the first $d'-1$ bits of the second node, which is the requirement for consistent choice of representative symbols. For example, a transition between a representative symbol $a_0a_1\dots a_{d'-1}$ in index i and a representative symbol $a_1a_2\dots a_{d'}$ in index $i+1$ is modeled by an edge from node ${}^0_1a_1\dots a_{d'-1}$ to node ${}^0_1a_2\dots a_{d'}$. The cost of the edge is $sr(i+1,a_1a_2\dots a_{d'}) \triangleq cost'(i+1,a_1a_2\dots a_{d'})$, where $cost'$ is folded using the folding property from $cost(i,x) \triangleq func_{i,s_1}[x] \triangleq \sum_{l_1} E_{l_1,i}[s'_1[l_1]\oplus x]$, as described in Section 4.3, and s'_1 is fixed for the given subgraph.

The total cost of edges along a path is $\sum_i func_{i,s_1}[s'_2[i]] = \sum_{l_1,i} E_{l_1,i}[s'_1[l_1]\oplus s'_2[i]] = score(s_1,s_2)$, where s'_2 is the candidate for S'_2 that is implied by the path, and s'_1 is the appropriate value for the j^{th} candidate for S_1. After a quick precomputation, the value of $func_{i,s_1}[x]$ can be calculated using a few table lookups regardless of the value of s_1.

5.1.2 The Structure of the Sub-graph Using Conditional Estimators

Similarly to the case of non-conditional estimators, in case conditional estimators are used, the subgraph for candidate j has a source node s_j, a target node t_j, and the subgraph is composed of several mini-subgraphs, which differ only in the location of the incoming edges (and their cost) and the location of the outgoing edge. However, with conditional estimators, the structure of the mini-subgraphs is different: each pair of indices $i, i+11$ are unified to a single index, denoted by $i|i+11$.

We would like to combine the nodes in index i with nodes in index $i+11$ by computing their cartesian product: for each node a in index i and for each node b in index $i+11$, we form the unified node $a|b$ in unified index $i|i+11$. However, there is a technical difficulty: while (given S_1) a non-conditional estimator depends on a symbol candidate $s'_2[i]$, a conditional estimator depends on both a symbol candidate $s'_2[i]$ and a clock symbol candidate $s_2[i+11]$. As a result, we must apply the D^{-1} operator on nodes in index $i+11$ (to transform them from symbols to clock symbols). This operation divides node $b = {}^0_1 b_1b_2\dots b_{d'-1}$ in index $i+11$ into two nodes ${}^0_1D_0^{-1}(b_1b_2\dots b_{d'-1})$ and ${}^0_1D_1^{-1}(b_1b_2\dots b_{d'-1})$. Only then, we can perform the cartesian product between the nodes in index i and the nodes that results from applying D^{-1}. Thus, from a pair of a and b of the above form, we have two nodes in the product (in index $i|i+11$): $a|{}^0_1D_0^{-1}(b_1b_2\dots b_{d'-1})$ and $a|{}^0_1D_1^{-1}(b_1b_2\dots b_{d'-1})$. We refer to the bits on the left of the "|" in the node as symbol bits, and the bits on the right of the "|" as clock bits. In total, there are $2^{d'-1}(2\cdot 2^{d'-1}) = 2^{2d'-1}$ nodes in each index $i|i+11$.

There is an edge from node $x_1|y_1$ in index $i|i+11$ to node $x_2|y_2$ in index $i+1|i+12$ if and only if the last $d'-1$ bits of x_1 are equal to the first $d'-1$ bits of x_2 and the last d' bits of y_1 are equal to the first d' bits of y_2. Figure 5 depicts four nodes of a mini-subgraph using conditional estimators.

What should be the cost of an edge? the basic cost function is $cost(i,x|y) \triangleq func_{i,s_1}[x|y] \triangleq \sum_{l_1} E_{l_1,i}[s'_1[l_1]\oplus x|s_1[l_1+10]\oplus y]$, which is folded to the cost

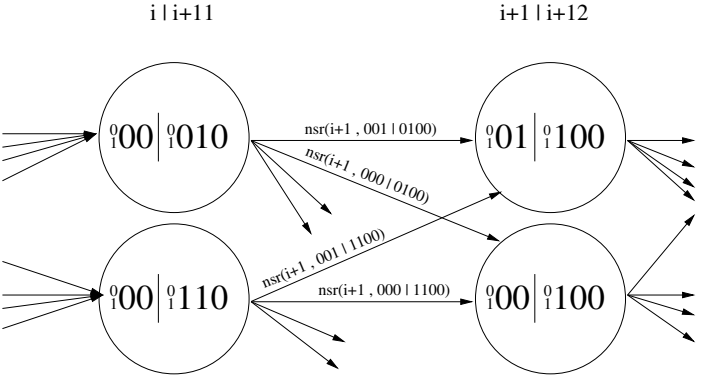

Fig. 5. Four nodes of the mini-subgraph using conditional estimators for $d' = 3$

function $cost'(i, x|y)$. Since each index $i|i+11$ unifies two indices, the edge that enters $i|i+11$ should contain the sum of contribution of indices i and $i+11$, i.e., the cost of the edge is $nsr(i, s_2'[i]|s_2[i+11]) \triangleq cost'(i, s_2'[i]||\mathrm{lsb}_{d'}(s_2[i+11])) + cost'(i+11, s_2'[i+11]|s_2[i+22])$, where $\mathrm{lsb}_{d'}(x)$ returns the d' first bits of x. Note that $s_2'[i+11] = D(s_2[i+11])$, and (due to the alignment property) $s_2[i+22] = s_2'[i]$. Therefore, $nsr(i, s_2'[i]|s_2[i+11]) = cost'(i, s_2'[i]||\mathrm{lsb}_{d'}(s_2[i+11])) + cost'(i+11, D(s_2[i+11])|s_2'[i])$.

Like the case of non-conditional estimators, we create several mini-subgraphs to ensure that the paths in the subgraph represent consistent choices for S_1 and S_2. We include in the subgraph a mini-subgraph for each combination v of the last $d'-1$ symbol bits and each combination w of the last d' clock bits of the last node (the node near t_j). A single edge (with cost zero) connects the mini-subgraph to t_j from node ${}^0_1v|{}^0_1w$. For consistency with the linear feedback, the bits w must be identical to the symbol bits of the first node (both w and the first symbol bits are d'-bit long). The bits v must be identical to the difference of the first d' bits of the first clock symbol. As v is $(d'-1)$-bit long, and as the clock bits of the first symbol are $(d'+1)$-bit long, there are four possibilities for the clock bits: $D_0^{-1}(v)||0$, $D_1^{-1}(v)||0$, $D_0^{-1}(v)||1$, and $D_1^{-1}(v)||1$. Therefore, four edges $w|D_0^{-1}(v)0$, $w|D_1^{-1}(v)0$, $w|D_0^{-1}(v)1$, and $w|D_1^{-1}(v)1$ connect s_j to the mini-subgraph (the concatenation mark "||" was removed for clarity). Their costs are $nsr(i_s, w|D_0^{-1}(v)0)$, $nsr(i_s, w|D_1^{-1}(v)0)$, $nsr(i_s, w|D_0^{-1}(v)1)$, and $nsr(i_s, w|D_1^{-1}(v)1)$, respectively.

To reconstruct s_2' from a path in the mini-subgraph, we first concatenate the symbol bits to form the first half of the path, and separately concatenate the clock bits to form the second half of the path. Then, we compute the difference between the clock bits, and combine the result with the symbol bits to obtain a path of s_2' (similar to the path in the case of the mini-subgraph using unconditional estimators).

Note that in an efficient implementation there is no need to keep the entire graph in memory, and needed parts of the graph can be reconstructed on-the-fly.

6 Simulations of Our Attacks

We have implemented our attack, and simulated it under various parameters. Our simulations focus on 2000 frames of data, which is the lowest amount of data that gives a non-negligible success rate in the simulations of Maximov, Johansson, and Babbage [11]. We also simulated the attack with 1500 frames. A comparison of simulations of previous attacks and simulations of our new attacks is given in Table 1.

In the simulations we use $d = 1$, $l_1 \in \{61, \ldots, 144\}$, $l_2 \in \{70, \ldots, 135\}$, and calculate estimators for $\mid l_1 - l_2 \mid < 10$. We use the first version of Step 3 with 64-bit keys.

We ran the simulations on a 1.8GHz Pentium-4 Mobile CPU with 512MB of RAM. The operating system was Cygwin under Windows XP. In comparison, the simulations of [11] were performed on a 2.4GHz Pentium-4 CPU with 256MB of RAM under Windows XP, and the simulations of [7] were performed on a 1.8GHz Pentium-4 CPU with 512MB of RAM under Linux.

In one simulation, we limited the size of the list of top (s_1, s_2) pairs to 5200. The key was found in about 64 percent of the cases, compared to about 5 percent in previous attacks with 2000 frames. Our attack takes about 7 seconds to complete Step 1. Step 2 takes about 340 seconds for the first pair, after which it can generate about 1500 pairs of candidates per second. Step 3 scans about 20.4 candidate pairs per second. Therefore, the total time complexity varies depending on the location of the correct pair in the list. It takes about 350 seconds (six minutes) in the best case, and up to ten minutes in the worst case.

For better results, we employ two methods: *early filtering* and *improved estimators*.

6.1 Early Filtering

In early filtering, we perform Step 2 several times, using less accurate (and faster) methods. Thus, we discard many candidate values of S_1 that are highly unlikely, and we do not need to build a subgraph for these values. For example, we score all the candidates of S_1 (a score of a candidate s_1 of S_1 is $\max_{s_2} score(s_1, s_2)$) using non-conditional estimators and a less accurate but faster method. Then, we recalculate the score for the 220000 top candidates, using a similar method, but with conditional estimators. The 40000 top scored candidates are re-scored using conditional estimators with a variation using only one mini-subgraph. Finally, we perform Step 2 of Section 5.1 with subgraphs only for the 2000 scored candidates of S_1. The list of the 5200 top candidates of S_1 and S_2 is generated and passed to Step 3. We denote this kind of configuration in a tuple $(220000, 40000, 2000, 5200)$. Simulation results using other configurations for both 2000 and 1500 frames are given in Table 1.

6.2 Improved Estimators

A disadvantage of the described attack is that only information from the estimators $E_{l_1,l_2}[\cdot|\cdot]$ is taken into consideration, while estimators involving $R3$, i.e.,

$E_{l_1,l_3}[\cdot|\cdot]$ and $E_{l_2,l_3}[\cdot|\cdot]$, are disregarded. In *improved estimators*, we improve our results by adding to each estimator $E_{l_1,l_2}[x|y]$ the contributions of the estimators of the other registers, i.e., we add to it

$$\sum_{l_3} \log \left(\sum_{\alpha,\beta \in \{0,1\}^d} e^{E_{l_1,l_3}[\alpha|\beta]+E_{l_2,l_3}[x\oplus\alpha|y\oplus\beta]} \right).$$

The resulting estimators include more information, and thus, are more accurate. They significantly improve the success rate with a modest increase in the time complexity of Step 1 (mostly, since we need to calculate three times the number of estimators). This increase in time complexity is compensated by a large decrease in the time complexity of Step 3 (as the correct S_1, S_2 are found earlier). The results are summarized in Table 1.

7 New Source for Known-Keystream

Every traffic channel between the handset and the network is accompanied by a slower control channel, which is referred to as the Slow Associated Control CHannel (SACCH). The mobile uses the SACCH channel (on the uplink) to report its reception of adjacent cells. The network uses this channel (on the downlink) to send (general) system messages to the mobile, as well as to control the power and timing of the current conversation.

The contents of the downlink SACCH can be inferred by passive eavesdropping: The network sends power-control commands to the mobile. These commands can be inferred from the transmission power of the mobile. The timing information that the network commands the mobile can be inferred from the transmission timing of the mobile. The other contents of the SACCH is a cyclical transmission of 2–4 "system messages" (see [8, Section 3.4.1]). These messages can be obtained from several sources, for example by passively eavesdropping the downlink at the beginning of a call (as the messages are not encrypted at the beginning of a call), or by actively initiating a conversation with the network using another mobile and recover these messages (these messages are identical for all mobiles). There is no retransmission of messages on the SACCH, which makes the task of the attacker easier, however, it should be noted that an SMS received during an on-going conversation could disrupt the eavesdropper, as the SMS can be transferred on the SACCH, when system messages are expected.

An attacker would still need to cope with the Frequency Hoping (FH) used by GSM. Using a frequency analyzer the attacker can find the list of n frequencies that the conversation hops on. Given n, GSM defines only $64n$ hopping sequences (n cannot be large since the total number of frequencies in GSM is only about 1000, of which only 124 belong to GSM 900). Thus, the hopping sequence can be determined through a quick exhaustive search.

As the name of SACCH implies, it is a slow channel. Only about eight frames are transmitted every second in each direction of the channel. Therefore, to collect 1500–2000 SACCH frames transmitted from the network to mobile, about 3–4 minutes of conversation are needed.

8 Summary

Our contribution in this paper is multi-faced. We begin by introducing conditional estimators that increase the bias of the correlation equation. Then, we present three weaknesses in $R2$, which were not reported previously. The first weakness — the alignment property — utilizes the fact that the correlation equation coincides with the feedback taps of $R2$. The second weakness — the folding property — uses the fact that $R2$ has only two feedback taps, and they are adjacent. We use the folding property to decode the estimators in an optimal way. In contrast, previous attacks were forced to use heuristics to decode the estimators. Using this weakness, we present a novel method to efficiently calculate the list of best candidate pairs for S_1 and S_2. Given S_1 and S_2, the value S_3 can be worked back from the keystream.

The last weakness that we report — the symmetry property — is based on the fact that $R2$'s clocking tap is exactly in its middle, which together with the folding property causes a symmetry between the clocking tap and the output of $R2$. This property enables us to efficiently decode the *conditional* estimators.

Finally, we describe a new source for known-plaintext in GSM. This source of known-plaintext transforms our attack to a practical ciphertext-only attack. With 3–4 minutes of raw ciphertext, we can extract (from the SACCH) the required amount of about 1500–2000 frames of known-plaintext.

We compare some of the previous results and our current simulation results in Table 1. Compared to previous attacks on 1500–2000 frames, it can be seen that our new attack has a significantly higher success rate (91% compared to 5%), it is faster, and it does not require any precomputation.

Acknowledgments

We are pleased to thank Alexander Maximov for providing early versions of [11].

References

1. Ross J. Anderson, *On Fibonacci Keystream Generators*, proceedings of Fast Software Encryption: Second International Workshop, Lecture Notes in Computer Science 1008, Springer-Verlag, pp. 346–352, 1995.
2. Edsger W. Dijkstra, *A Note on Two Problems in Connexion with Graphs*, Numerische Mathematik, Vol. 1, pp. 269–271, 1959.
3. Elad Barkan, Eli Biham, Nathan Keller, *Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communications*, Advances in Cryptology, proceedings of Crypto'03, Lecture Notes in Computer Science 2729, Springer-Verlag, pp. 600–616, 2003.
4. Eli Biham, Orr Dunkelman, *Cryptanalysis of the A5/1 GSM Stream Cipher*, Progress in Cryptology, proceedings of Indocrypt'00, Lecture Notes in Computer Science 1977, Springer-Verlag, pp. 43–51, 2000.
5. Alex Biryukov, Adi Shamir, David Wagner, *Real Time Cryptanalysis of A5/1 on a PC*, Advances in Cryptology, proceedings of Fast Software Encryption'00, Lecture Notes in Computer Science 1978, Springer-Verlag, pp. 1–18, 2001.

6. Marc Briceno, Ian Goldberg, David Wagner, *A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms*, `http://cryptome.org/gsm-a512.htm` (originally on www.scard.org), 1999.
7. Patrik Ekdahl, Thomas Johansson, *Another Attack on A5/1*, IEEE Transactions on Information Theory, Volume 49, Issue 1, pp. 284-289, 2003.
8. European Telecommunications Standards Institute (ETSI), *Digital cellular telecommunications system (Phase 2+); Mobile radio interface; Layer 3 specification*, TS 100 940 (GSM 04.08), `http://www.etsi.org`.
9. Jovan Golic, *Cryptanalysis of Alleged A5 Stream Cipher*, Advances in Cryptology, proceedings of Eurocrypt'97, LNCS 1233, pp. 239–255, Springer-Verlag, 1997.
10. Walter Hoffman, Richard Pavley, *A Method for the Solution of the Nth Best Path Problem*, Journal of the ACM (JACM), Volume 6, Issue 4, pp. 506–514, 1959.
11. Alexander Maximov, Thomas Johansson, Steve Babbage, *An improved correlation attack on A5/1*, proceedings of SAC'04, LNCS 3357, pp. 1–18, Springer-Verlag, 2005.
12. Willi Meier, Othmar Staffelbach, *Fast Correlation Attacks on Certain Stream Ciphers*, Journal of Cryptology, Volume 1, Issue 3, pp. 159–176, Springer-Verlag, 1989.
13. Thomas Siegenthaler, *Decrypting a Class of Stream Ciphers Using Ciphertext Only*, IEEE Transactions on Computers, Volume 49, Issue 1, pp. 81–85, 1985.