

An Improved Guess-and-Determine Attack on the A5/1 Stream Cipher

Ayan Mahalanobis¹ & Jay Shah¹

¹ IISER Pune, Dr. Homi Bhabha Road, Pashan Pune 411008, India

Correspondence: Ayan Mahalanobis, IISER Pune, Dr. Homi Bhabha Road, Pashan Pune 411008, India. E-mail: ayan.mahalanobis@gmail.com

Received: November 3, 2013 Accepted: December 8, 2013 Online Published: January 13, 2014

doi:10.5539/cis.v7n1p115 URL: <http://dx.doi.org/10.5539/cis.v7n1p115>

Abstract

In Europe and North America, the most widely used stream cipher to ensure privacy and confidentiality of conversations in GSM mobile phones is the A5/1. In this paper, we present an improved *guess-and-determine attack* on the A5/1 stream cipher with an average time complexity of $2^{48.5}$, which is much less than any known guess-and-determine attack. The attack has a 100% success rate and requires a small amount of memory. We provide a detailed description of our new attack along with its implementation results.

Keywords: A5/1, GSM, guess-and-determine attack, stream ciphers

1. Introduction

The most widely used stream cipher to ensure privacy and confidentiality of communications in GSM mobile phones in Europe and North America is the A5/1. The A5/1 was developed in 1987 when GSM was not considered for use outside of Europe. The description of the A5/1 was initially kept secret. However, its design was disclosed in 1999 by reverse engineering (Briceno, Goldberg, & Wagner, 1999). The GSM organization later confirmed the disclosed algorithm (Biryukov, Shamir, & Wagner, 2001).

1.1 Our Contributions

Broadly speaking, attacks on the A5/1 can be classified into known-plaintext attacks and time-memory trade-off attacks. There are some exceptions to this: for example, Ekdahl and Johansson (2003) produced an attack that exploits bad key initializations in A5/1. Other such examples are the *bounded distance decoder* (BDD) attack (Krause, 2002) or a ciphertext only attack by Barkan et al. (2008). However BDD attack is clearly exponential in the length of the shift registers. There is a general criticism against the time-memory trade-off attacks, they are exponentially more expensive. So one can just increase the register lengths to avoid such attacks.

In this paper, we describe an improved *guess-and-determine* attack on the A5/1 stream cipher. This attack has an average complexity of $2^{48.5}$ steps, and is better than all known guess-and-determine attacks and expands on a novel idea. Our attack is simple to describe and easy to analyze. Guess-and-determine attacks are of interest because of many reasons; three most important ones are:

- (a) They are easy to implement, much easier than the time memory trade-off attacks.
- (b) They can be efficiently implemented in parallel programming environment.
- (c) They are easy to describe.

Our attack is a *known-plaintext attack*. It can be briefly described (Note 1) as follows, (ref. Figure 1): we assume that the register R_1 is full (guessed) with 19 bits and registers R_2 and R_3 will be filled (determined) sequentially as the attack progresses. At any stage of this attack, R_1 is completely filled and R_2 and R_3 are partially filled. We call these states as **state candidates**. Once all three registers are completely filled, we call that state candidate a **complete state candidate**. Our attack has a 100% success rate and has low memory requirement. With the knowledge of only 11 bits of the known keystream, the attack algorithm is able to determine a set of *complete state candidates* which may contain the key. With every additional round of the attack, the number of complete state candidates increase. We provide a detailed description of our new attack along with its implementation in Sections

4 and 5.

The **complexity of this attack** is about $2^{48.5}$ A5/1 clockings when done in serial. This means that we go over the guesses of R_1 one after another. However one can easily parallelize this, each thread of computing starts with an independent and different guess. In this case the complexity is substantially reduced. In case of the extreme situation, when we start with 2^{19} threads of computation, the complexity is $2^{29.5}$.

Let us say this upfront: we were unable to do a large-scale industrial-grade implementation of this attack because of resource constraints. We did an implementation, which was not fully optimized. We report some of those results in Table 1.

2. A Brief Description of the A5/1 Stream Cipher

The A5/1 stream cipher is built from three short *linear feedback shift registers* (LFSR) of lengths 19, 22 and 23 bits. We denote these by R_1 , R_2 and R_3 respectively. The rightmost bit in each register is labeled as bit zero. The tapping bits of R_1 are at bit positions 13, 16, 17, 18, of R_2 are 20 and 21, and of R_3 are 7, 20, 21 and 22 (ref. Figure 1).

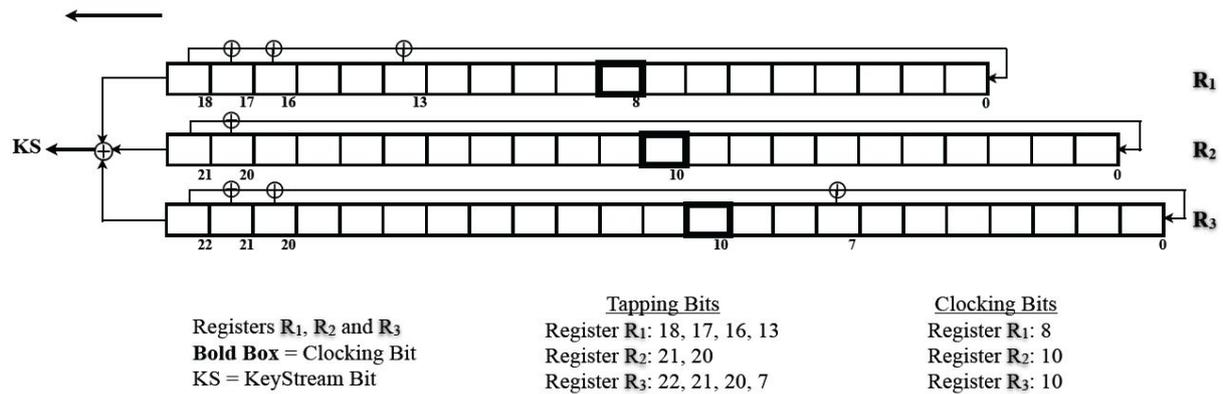


Figure 1. A5/1 stream cipher

The A5/1 keystream generator works as follows: First, an *initialization phase* is run giving rise to a initial state. Based on this initial state, a warm-up phase is performed. In the keystream production stage the registers are clocked in a stop-and-go fashion using the following **majority rule**. Each register has a single *cloning bit* (bit 8 for R_1 , 10 for R_2 , and 10 for R_3) which decides the clocking pattern for that register. Before each clocking cycle, the cloning bits are observed; they are either 0 or 1. The majority function outputs the registers that have the most similar bits. Those registers are clocked. At each step either two or three registers are clocked, and each register has 3/4 probability of moving.

A total of four clocking pattern are possible. They are:

- $CB_1 = CB_2 \neq CB_3$ (Clock only R_1 and R_2)
- $CB_1 \neq CB_2 = CB_3$ (Clock only R_2 and R_3)
- $CB_1 = CB_3 \neq CB_2$ (Clock only R_1 and R_3)
- $CB_1 = CB_2 = CB_3$ (Clock all three registers)

where CB_i denotes the cloning bit for register i where $i \in \{1, 2, 3\}$.

After clocking, an output bit is generated from the values of R_1 , R_2 , and R_3 by XORing their *most significant bits*, as shown in Equation 1. This XORed bit is called the *keystream bit*. We denote the full keystream by KS and the i^{th} bit by $KS[i]$. The keystream is a sequence of these keystream bits indexed by the clocking, where every clock produces a keystream bit.

$$R_1[18] \oplus R_2[21] \oplus R_3[22] = \text{a keystream bit.} \tag{1}$$

3. Known Attacks on the A5/1

This section surveys many known *guess-and-determine* attacks on the A5/1. A guess-and-determine attack is a

known-plaintext attack on a stream cipher, where the attacker knows some bits of the keystream and the remaining bits are determined from the known keystream bits. A known plaintext attack, is an attack model where the attacker has access to both the plaintext and its encrypted ciphertext. This can be used to reveal the keystream used for encrypting the known plaintext to the ciphertext. Guess-and-determine attacks include Anderson's attack (Anderson, 1994), Golic's attack (Golic, 1997), Biham-Dunkelman's attack (Biham & Dunkelman, 2000), Keller-Seitz's attack (Keller & Seitz, 2001) and Gendrullis-Novotny-Rupp's attack (also known as the modified Keller-Seitz attack) (Gendrullis, Novotny, & Rupp, 2008). All these attacks assume that consecutive 64 bits of the keystream are known.

3.1 Guess-and-Determine Attacks

The first guess-and-determine attack on the A5/1 was proposed by Anderson (1994). Anderson suggested guessing all bits of registers R_1 and R_2 and the lower half of register R_3 (i.e., $19 + 22 + 11 = 52$ bits) to determine the remaining bits of R_3 by Equation 1. In the worst-case, each of the possible 2^{52} state candidates would have to be verified against the known keystream. This attack was not implemented as Biham-Dunkelman's attack and Keller-Seitz's attack had better complexity.

Golic proposed an attack that had a complexity of 2^{40} , additionally one has to solve a 64×64 set of linear equations (Golic, 1997). His approach was to guess the lower half of all three registers and determine the remaining bits of the registers with the known keystream by Equation 1. However, each operation in this attack was much more complicated as it was based on finding solutions of a system of linear equations. In practice, Anderson's approach (Anderson, 1994) and Keller-Seitz's (Keller & Seitz, 2001) approach are better than Golic's attack.

Pornin and Stern (2000) proposed a *software-hardware trade-off* attack, which was based on Golic's approach. But in contrast to Golic's approach, they guessed the clocking sequence at the very beginning. The increased assumptions and complexity of the attacks made the actual implementation very difficult and impractical.

The Biham-Dunkelman attack (Biham & Dunkelman, 2000) was expected to be a thousand times faster than the Anderson's attack (Biham & Dunkelman, 2000) or Keller-Seitz's attack (Keller & Seitz, 2001). The attack requires 2^{47} A5/1 clockings and about $2^{20.8}$ bits of plaintext data, which is equivalent to 2.36 minutes of conversation. The attacker guesses 12 bits from $R_1[9]$ to $R_1[12]$, from $R_1[14]$ to $R_1[18]$, $R_2[0]$, $R_3[22]$ and $R_3[10]$, and determines the remaining bits of registers R_1 and R_2 by Equation 1 and the known keystream bits. The attack algorithm assumes that register R_3 is not clocked (i.e., $R_1[8] = R_2[10] \neq R_3[10]$) for 10 consecutive rounds. Such an event will occur in one out of 2^{20} possible cipher states. The attacker must know the exact location of the information-leaking event where register R_3 is not clocked for 10 consecutive rounds. This is a big assumption. Thus, the attacker will need to probe about 2^{20} different starting locations by trial-and-error before the event actually occurs. This attack requires a lot of data and precomputation space. Hence this attack is not practical for implementation.

Keller and Seitz designed a new attack (Keller & Seitz, 2001) based on the attack proposed by Anderson. But unlike Anderson's approach, they took into account the asynchronous clocking of the A5/1 stream cipher. According to their algorithm, the attacker guesses registers R_1 and R_2 completely and determines all bits of register R_3 by Equation 1. The attack was divided into two phases: a *determination phase* in which a possible state candidate consisting of the three registers of A5/1 after its *warm-up phase* (Briceno et al., 1999) is generated, and a subsequent *post-processing-phase* in which the state candidate is checked for consistency. In the determination phase, the authors try to reduce the complexity of the simple guess-and-determine attack by early recognizing contradictions that could occur on guessing the clocking bit of R_3 such that R_3 will not be clocked. Hence, all states arising out of the contradictory guesses neither need to be computed further nor checked afterwards. The authors not only discard the incorrect possibilities for $R_3[22]$ in case of contradiction, but also limit the number of choices to the one of non-clocking R_3 , when this is possible without any contradiction. This further reduces the complexity. If a case arises where $R_1[8] = R_2[10]$ and $R_3[10]$ has to be guessed, the authors suggest to always consider the case $R_1[8] = R_2[10] = R_3[10]$ and clock register R_3 with register R_1 and register R_2 . This leaves out the possible case of $R_1[8] = R_2[10] \neq R_3[10]$. Thus, the success probability of this attack is approximately 18%, and the number of state candidates inspected by Keller and Seitz to the number of valid states is $\frac{86}{471} \approx 0.18$.

Gendrullis, Novotny and Rupp (2008) (GNR) proposed a modification to the Keller-Seitz attack. Unlike Keller-Seitz (Keller & Seitz, 2001), the authors only discard the wrong possibilities for the clocking bit of register R_3 which would lead to a contradiction. But if no contradiction exists, they consider both cases: clocking and not-clocking of R_3 . Thus, every possible state candidate is taken into account. This gives us a success probability of 100%. The attacker needs an expected 17.67 clocking rounds to determine a complete state candidate and check it

for consistency with the given keystream. The time complexity of the complete attack is $2^{54.02}$.

Besides Golic (1997) and Babbage (1995), Biryukov, Shamir and Wagner (2001) proposed an attack with a complexity of 2^{48} , which requires about 300 GB storage, where the online phase of the attack can be executed within minutes and has a success probability of 60%.

Barkan-Biham-Keller et al. (2008) also proposed another attack along these lines. However, in the precomputation phase of such an attack huge amount of data need to be computed and stored. For example, with three minutes of ciphertext available, one needs to precompute about 50 TB of data to achieve a success probability of about 60%. These are practical obstacles that make the implementation of such attacks very difficult.

4. An Improved Guess-and-Determine Attack on the A5/1 Stream Cipher – Our Attack

Our approach is based on the *guess-and-determine* attack proposed by Anderson (1994), but with novel modifications that makes the attack faster. With 64 bits of the keystream known, all bits of register R_1 are guessed and all bits of registers R_2 and R_3 are determined. Eventually, we have about $2^{48.5}$ possible state candidates, which is better than all known guess-and-determine attacks, see Table 2 for details.

The attack consists of two phases, the **determination phase** and the **post-processing phase**. The *determination phase* is again divided into two parts, the **processing-phase1** and the **processing-phase2**.

4.1 Determination Phase

We assume that the register R_1 is full and the registers R_2 and R_3 are vacant. We are trying to fill these two registers in this phase with the help of a known keystream. We introduce *two counters* t_2 and t_3 and initialize them to 0. Every time register R_2 moves we increment the counter t_2 by one and similarly for R_3 we increment t_3 .

4.1.1 Processing-Phase1

Compute the most significant bits of register R_2 and register R_3 using the MSB of register R_1 and KS bit by Equation 1. If the values of three of these bits are known, the fourth can be computed easily by the equation. If $R_2[21]$ and $R_3[22]$ are unknown, then there exist four possible combinations for the unknown bits – 00, 01, 10 and 11. But Equation 1 reduces the number of possibilities to two. The two possible combinations that satisfy the equation are:

- If $R_1[18] = \text{KS bit}$, then $R_2[21] = R_3[22] = 0$ or $R_2[21] = R_3[22] = 1$.
- If $R_1[18] \neq \text{KS bit}$, then $R_2[21] = 0, R_3[22] = 1$ or $R_2[21] = 1, R_3[22] = 0$.

This reduces the number of possible cases by half and the number of possible state candidates to two. For more details see Figure 2.

4.1.2 Processing-Phase2

Consider the clocking bits of registers R_2 and R_3 . There are three possibilities:

- If $R_2[10]$ is filled and $R_3[10]$ is vacant, then replicate the state candidate twice, fill one copy with $R_3[10] = 0$, and the other copy with $R_3[10] = 1$.
- If $R_2[10]$ is vacant and $R_3[10]$ is filled, then replicate the state candidate twice, fill one copy with $R_2[10] = 0$, and the other copy with $R_2[10] = 1$.
- If $R_2[10]$ and $R_3[10]$ are both vacant, then replicate the state candidate four times, fill the first copy with $R_2[10] = 0, R_3[10] = 0$; the second copy with $R_2[10] = 0, R_3[10] = 1$; the third copy with $R_2[10] = 1, R_3[10] = 0$; and the fourth copy with $R_2[10] = 1, R_3[10] = 1$.
- Since the bit $R_3[7]$ is a feedback bit, we need to take special care of $R_3[7]$. If there is a feedback, i.e., clocking in R_3 , that bit needs to be full. So after each replication we see whether there will be clocking in the register R_3 from the majority function. If there is clocking and $R_3[7]$ is vacant, we duplicate that state candidate and fill 0 in $R_3[7]$ for one, and 1 in the other.

Thus, all possible combinations are taken into consideration. Further details are available in Figure 3.

Now consider the bits $R_2[20]$ and $R_3[21]$ and let $\text{KS}[i]$ be the known keystream bit for some $i \in \mathbb{N}$. If registers R_2 and R_3 are clocked, then these bits will become the new MSBs for their respective registers after clocking. If both these bits are vacant, there are four possible combinations for these bits; i.e., 00, 01, 10 and 11. But Equation 4.1 in Figure 3 ($R_1[17] \oplus \text{KS}[i+1] = R_2[20] \oplus R_3[21]$) and Equation 4.2 ($R_1[18] \oplus \text{KS}[i+1] = R_2[20] \oplus R_3[21]$) reduce

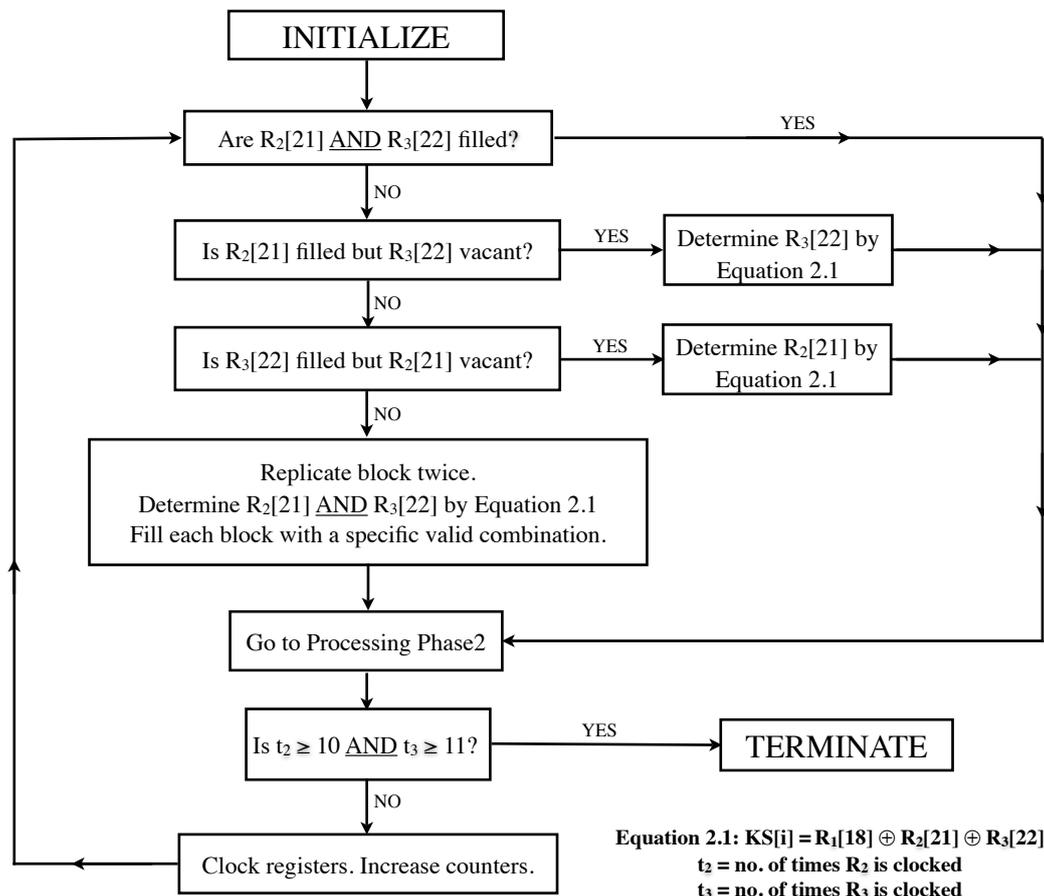


Figure 2. Determination phase of the attack (processing-phase1)

them to two possibilities. This reduces the number of possible cases by half, a 50% save.

If only one of these bits is vacant, there are two possibilities for the vacant bit, 0 or 1. But this is reduced to only one possibility. For example, if $R_2[10] \neq R_1[8] = R_3[10]$, then $R_3[21] = R_1[17] \oplus R_2[21] \oplus KS[i + 1]$. In this case, only $R_3[21]$ is unknown. This bit can be calculated by the above equation. Here, two possibilities for $R_3[21]$ reduce to only one possibility. This reduces the number of cases by half.

Follow this protocol as long as $t_2 < 10$ or $t_3 < 11$. When this condition is not satisfied, i.e., the first time $t_2 \geq 10$ and $t_3 \geq 11$, stop. At this moment, registers R_2 and R_3 are completely determined for the known KS and register R_1 . The number of bits between the clocking bit (CB) and the MSB for register R_2 is 10 and for register R_3 is 11. Hence, register R_2 has to be clocked at least 10 times and register R_3 has to be clocked at least 11 times to determine all the bits in those registers.

A *complete state candidate* is a state candidate with all bits filled. The minimum number of KS bits required to obtain a set of complete state candidates is eleven. This will happen when both registers R_2 and R_3 are clocked together for 10 consecutive clocking cycles and register R_3 is clocked again in the next round.

4.2 Post-Processing-Phase

The post-processing-phase checks for the key from the set of complete state candidates obtained after the determination phase. As discussed in Section 4.1, the minimum number of rounds needed to perform the post-processing-phase is 11. The number of complete state candidates increases with every additional round. Hence, the probability of finding the key increases with every additional round.

In this phase, we generate output bits by performing normal A5/1 encryption with each of the complete state candidates obtained from the *determination phase*. Match these output bits bit-wise with the known KS bits. If the KS bits and output bits match, continue clocking and generating output bits until a contradiction of bit-wise

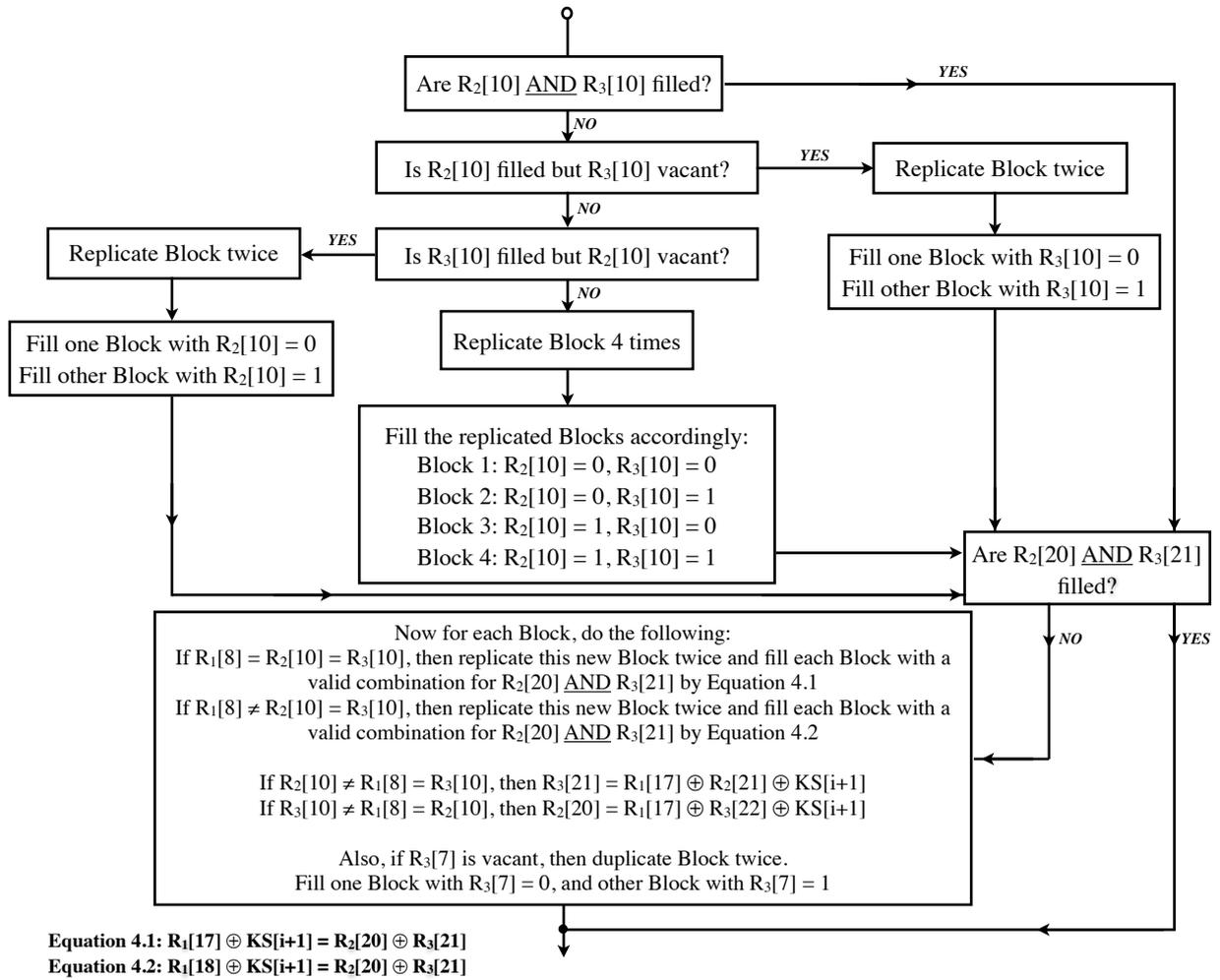


Figure 3. Determination phase of the attack (processing-phase2)

matching occurs. If all the output bits match the given 64 KS bits, that complete state candidate is the key. We have thus found the key among the set of complete state candidates.

5. Analysis of the Attack

After initialization, we perform the determination phase. At this stage, all bits of register R_1 are filled, while registers R_2 and R_3 are vacant. According to the attack protocol, the determination phase determines the most significant bits of registers R_2 and R_3 ($R_2[21]$ and $R_3[22]$) in the processing-phase1; the clocking bits of R_2 and R_3 ($R_2[10]$ and $R_3[10]$), if possible bit $R_3[7]$, bits $R_2[20]$ and $R_3[21]$ by processing-phase2.

If vacant, MSB of registers R_2 and R_3 are to be determined. The number of possible combinations reduces from four to two by Equation 1. During the implementation of further rounds, there may be a possibility where only one of the MSB of R_2 or R_3 is vacant. Theoretically, the number of combinations to fill that vacant bit is two, but Equation 1 reduces that to only one correct possibility.

Processing-phase2 of the determination phase considers four bits: $R_2[10]$ (CB of R_2), $R_3[10]$ (CB of R_3), $R_2[20]$ and $R_3[21]$. All these four bits may or may not be vacant at all times. In the following table (Figure 4), we consider all possible cases of these four bits being vacant or full, and the number of maximum possible valid combinations that exist as a result of Equation 1. The last column depicts the percentage of the total possible cases that are discarded due to the attack algorithm compared to an exhaustive search.

Now let us consider the bit $R_3[7]$ in register R_3 , which is part of the feedback bit for the register. If this bit is vacant, there are two possibilities, 0 or 1. We could not eliminate any of these possibilities in our algorithm, so this case is not included in Figure 4.

EMPTY?				POSSIBLE CASES	MAX. POSSIBLE VALID CASES	% SAVE
CB2	R ₂ [20]	CB3	R ₃ [21]			
✓	✓	✓	✓	16	6	62.5
✓	✓	✓	-	8	NA	NA
✓	✓	-	✓	8	3	62.5
✓	-	✓	✓	8	4	50
-	✓	✓	✓	8	3	62.5
✓	✓	-	-	4	2	50
✓	-	✓	-	4	NA	NA
✓	-	-	✓	4	2	50
-	✓	-	✓	4	2	50
-	✓	✓	-	4	NA	NA
-	✓	-	✓	4	2	50
-	-	✓	✓	4	2	50
✓	-	-	-	2	2	0
-	✓	-	-	2	1	50
-	-	✓	-	2	NA	NA
-	-	-	✓	2	1	50
-	-	-	-	0	0	0

Figure 4. All possibilities during Processing-Phase2

After the determination phase of our algorithm, if the clocking bit of R_3 is vacant, the bit $R_3[21]$ must also be vacant. It is impossible to have a case where clocking bit of R_3 is vacant, but bit $R_3[21]$ is filled. This reduces the number of possible valid cases and is denoted in Figure 4 as *Not Applicable* (NA) cases.

In the determination phase, a total of 7 bits (i.e., $R_2[21]$, $R_2[20]$, $R_2[10]$, $R_3[22]$, $R_3[21]$, $R_3[10]$ and $R_3[7]$) have to be determined. These 7 bits would have $2^7 = 128$ possible combinations. But the attack algorithm gives only 24 valid possible combinations. Thus rejecting 104 combinations, a saving of 81.25%.

If $R_3[7]$ is not considered, the first round of implementation will always generate 12 state candidates. On an average, the second round generates 60 state candidates and the third round generates 300 state candidates. The number of state candidates (up to round 10) can be approximated by the formula $12 \times 5^{n-1}$, where n denotes the number of round, $1 \leq n < 11$. It is only after the 11th round that we get the first set of complete state candidates. When the bit $R_3[7]$ is taken into consideration, the first round of implementation will always generate 24 state candidates. From round three to round ten, the number of possible state candidates after every round is approximately five times the total number in the previous round.

6. Discussion

We discuss in detail a probabilistic approach to determine the time complexity and success probability of our new attack. The results of this probabilistic approach are also corroborated by experimental data. According to these results, the average number of rounds necessary to get the key is 15.5 and the average number of complete state candidates obtained after 15.5 rounds is $2^{48.5}$. We conclude this section with a comparison of our attack with other known attacks.

6.1 Time Complexity

We now study the time-complexity of our algorithm. The work done by this algorithm can be split in the following:

- checking to see if a cell in a register is vacant or full, in this paper we often called it as a bit is full or vacant.
- replication of state candidates.
- filling up vacant bits.

It is reasonable to assume that the checking and filling of bits take negligible amount of time. Hence, we can safely assume that the unit of our time complexity measurement should be the number of replications needed, where one unit of time is one replication. The algorithm starts with registers R_2 and R_3 vacant and register R_1 filled (guessed). At the end, it creates about $2^{48.5}$ complete state candidates, i.e., $2^{48.5}$ replications.

The number of bits between the clocking bit (CB) and the most significant bit for register R_2 is 10 and for register R_3 is 11. Hence, the number of times the registers R_2 and R_3 have to be clocked to fill all the bits is at least 10 and 11 respectively. The minimum number of KS bits required to obtain a set of complete state candidates is 11. This will occur when both registers R_2 and R_3 are clocked together for 10 consecutive clocking cycles and register R_3 is clocked again in the following round. For one guess of the register R_1 , the number of complete state candidates after 11 clocking rounds is approximately 2^{40} . With every clocking round, the number of complete state candidates increases.

According to the *majority function* for the clocking rule of the A5/1, a register will get clocked 3 out of 4 times. At every clocking cycle, at least two registers will get clocked. As stated in Section 2, a total of four cases are possible for the clocking patterns of the registers and each are equally likely with a probability of 0.25. Let n_1 be the event that exactly one of the registers R_2 or R_3 is clocked along with R_1 . Let n_2 be the event that both the registers R_2 and R_3 are clocked. The probability that an event n_i to occur is denoted by $P(n_i)$ where $i = 1, 2$. Thus $P(n_i) = 0.5$ for each i . Registers R_2 and R_3 have to be clocked at least 10 and 11 times respectively to determine all bits in those registers.

Let X be the random variable denoting the number of clocking cycles needed to obtain a complete state candidate. Let x_1 be the minimum number of clocking cycles needed for event n_1 to get a complete state candidate. Let x_2 be the number of clocking cycles needed to get a complete state candidate from the event n_2 . It is easy to see that $x_1 = 21$, $x_2 = 10$. The expectation for this variable X is given by

$$\begin{aligned} E[X] &= x_1 \times P(n_1) + x_2 \times P(n_2) \\ &= 21 \times \frac{1}{2} + 10 \times \frac{1}{2} = 15.5 \end{aligned}$$

We concluded earlier that the minimum number of clocking cycles necessary to obtain a set of complete state candidates is 11. After 15.5 rounds, there is a very high probability that the set of complete state candidate contains the key. Experimental results show us that after 11 rounds we get about 2^{40} complete state candidates. The advantage of this attack is that once we have a set of complete state candidates, we can perform the post-processing-phase separately and simultaneously, independent of the processing-phases of the next round and save time.

6.2 Success Probability

In Table 1, we describe the data obtained from our experiments with this attack. The four columns of the table are: number of clocking rounds, total number of state candidates obtained after that round, total number of complete state candidates obtained, and the percentage of complete state candidates over the total number of state candidates for that particular round. All values of the experimental data in the table are approximated to one decimal place.

Table 1. Storage requirement and success probability

No. of Rounds	Total State Candidates	Complete State Candidates	$\frac{\text{Complete}}{\text{Total}} \times 100$
11	$2^{45.2}$	$2^{39.2}$	1.6%
12	$2^{46.0}$	$2^{42.5}$	9.0%
13	$2^{46.7}$	$2^{44.5}$	22%
14	$2^{46.9}$	$2^{45.3}$	30%
15	$2^{47.1}$	$2^{46.1}$	50%

Remark: Our guess-and-determine attack guesses the entries of the register R_1 and tries to determine the bits of the register R_2 and R_3 . Once R_1 is fixed we have 19 fixed bits. In Figure 2, and the subsequent text we produced an algorithm to determine these bits. We argued that the number of choices our algorithm makes is at most half the amount of choices one would have made, if all possible combinations were taken into account. We also showed that one would need, in average, about 15.5 clocking cycles. Now if all the possible choices of R_2 and R_3 were considered then that would be 2^{45} choices. In 15.5 clock cycles we would fill the registers R_2 and R_3 if all possible choices were made. So the number of choices in our algorithm is $2^{45} \times \left(\frac{1}{2}\right)^{15.5} = 2^{29.5}$. So for each possible choices of the register R_1 there are $2^{29.5}$ choices. There are 2^{19} choices for the register R_1 . If one works in serial, one choice for R_1 after another then the total number of A5/1 clocking is $2^{19} \times 2^{29.5} = 2^{48.5}$. It is now clear that this complexity can be substantially reduced by parallelizing this algorithm, where each thread of computation takes on a different choice for R_1 .

Table 2. Comparison of the known attacks on the A5/1

Attack	KS bits	Time complexity	Success probability	Notes/Storage
Anderson (Anderson, 1994)	64	2^{52}	100%	–
Golic (Golic, 1997)	64	$2^{40.16}$	100%	additionally solve 64x64 linear system of equations
Biham-Dunkelman (Ghafari & Mohajeri, 2011)	–	–	–	–
(Ghafari & Mohajeri, 2011)	$2^{20.8}$	$2^{37.89}$	63%	64 GB Storage
Keller-Seitz (Keller & Seitz, 2001)	64	$2^{51.24}$	18%	–
GNR (Gendrullis et al., 2008)	64	$2^{54.02}$	100%	–
BSW (Biryukov et al., 2001)	64	2^{48}	60%	300 GB
Rainbow Table (Nohl, 2010)	64	few seconds	90%	2 TB storage and 2 GPU
Our Attack	64	$2^{48.5}$	100%	–

References

- Anderson, R. (1994). *A5 (was: Hacking digital phones)*. Newsgroup Communication, 1994.
- Babbage, S. (1995). A space time tradeoff in exhaustive search attacks on stream ciphers. In *European convention on security and detection*. <http://dx.doi.org/10.1049/cp:19950490>
- Barkan, E., Biham, E., & Keller, N. (2008). Instant ciphertext-only cryptanalysis of GSM encrypted communication. *Journal of Cryptology*, 21(3), 392-429. <http://dx.doi.org/10.1007/s00145-007-9001-y>
- Biham, E., & Dunkelman, O. (2000). Cryptanalysis of the A5/1 GSM stream cipher. In *Indocrypt'00*.
- Biryukov, A., Shamir, A., & Wagner, D. (2001). Real time cryptanalysis of A5/1 on a PC. In *FSE'00*.
- Briceno, M., Goldberg, I., & Wagner, D. (1999). *A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms*. Retrieved January 5, 2012, from <http://cryptome.org/gsm-a512.htm>
- Ekdahl, P., & Johansson, T. (2003). Another attack on A5/1. *IEEE Transactions on Information Theory*, 49(1), 284-289. <http://dx.doi.org/10.1109/TIT.2002.806129>
- Gendrullis, T., Novotny, M., & Rupp, A. (2008). A real-world attack breaking A5/1 within hours. In *CHES'08* (pp. 266-282).
- Ghafari, V., & Mohajeri, J. (2011). An improved attack on A5/1. In *Information security and cryptology* (pp. 41-44).
- Golic, J. (1997). Cryptanalysis of alleged A5 stream cipher. In *Eurocrypt'97* (pp. 239-255).
- Keller, J., & Seitz, B. (2001). *A hardware-based attack on the A5/1 stream cipher (Tech. Rep.)*. FernUniversitat in

- Hagen, Germany*. Retrieved June 14, 2011, from <http://pv.fernuni-hagen.de/docs/apc2001-final.pdf>
- Krause, M. (2002). BBD-based cryptanalysis of key stream generators. In *Eurocrypt'02* (pp. 222-237).
- Nohl, K. (2010). *Attacking phone privacy*. (Blackhat 2010 Lecture Notes)
- Pornin, T., & Stern, J. (2000). Software-hardware trade-offs: Application to A5/1 cryptanalysis. In *CHES'00* (pp. 318-327).

Notes

Note 1. A reader not familiar with the A5/1 stream cipher is encouraged to read Section 2 first.

Copyrights

Copyright for this article is retained by the author(s), with first publication rights granted to the journal.

This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).