A New Guess-and-Determine Attack on the *A5/1*

Jay Shah and Ayan Mahalanobis

Indian Institute of Science Education and Research, Pune, India
`jayshah_89@hotmail.com; ayan.mahalanobis@gmail.com`

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 a new attack on the *A5/1* stream cipher with a minimum time complexity of around 2^{40} and an average complexity of $2^{48.5}$, which is much less than the brute-force attack with a complexity of 2^{64}. The attack has a 100% success rate and requires about 5.65GB storage. We provide a detailed description of our new attack along with its implementation and 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 conversations in GSM mobile phones in Europe and North America is the *A5/1*. The *A5/1* was developed in 1987, when GSM was not yet considered for use outside Europe. The description of the *A5/1* was initially kept secret, but it's design was disclosed in 1999 by reverse engineering [6]. The GSM organization has later confirmed the correctness of the algorithm [5].

There are multiple versions of the encryption algorithm which belong to the *A5* family: *A5/0* is a dummy cipher with no encryption; *A5/1* is the original *A5* algorithm ensures over-the-air communication privacy and confidentiality of conversations in GSM mobile phones; *A5/2* is an intentionally weaker encryption algorithm created for export; while *A5/3* is a strong encryption algorithm created as part of the 3rd Generation Partnership Project *(3GPP)* which is currently responsible for maintaining and developing GSM technical specifications around the world [12].

Anderson [1], Golic [11] and Babbage [2] were the pioneers in initially cryptanalyzing the *A5/1* encryption algorithm when only a rough outline of the *A5/1* was leaked. After *A5/1* was reverse engineered, it was analyzed by Biryukov, Shamir and Wagner [5]; Biham and Dunkelman [4]; Ekdahl and Johansson [8]; Maximov, Johansson and Babbage [15]; Barkan and Biham [3]; Keller and Seitz [13]; and a few other researchers.

1.1 Current Research

Several attacks on the *A5/1* stream cipher have been designed in the past twenty years, but only a few of them have been implemented. Attacks on the GSM protocol can work even if the network supports only *A5/1* or *A5/3* encryption, as long as the mobile phone supports *A5/2* encryption. The main flaw that allows the implementation of these attacks is that the same key is used regardless of whether the phone encrypts using *A5/1*, *A5/2*, or

A5/3 algorithm. Therefore, the attacker can mount a man-in-the-middle attack, in which the attacker impersonates the mobile to the network, and the network to the mobile (by using a fake base station). The attacker might use *A5/1* for communication with the network and *A5/2* for communication with the mobile. But due to the flaw, both algorithms encrypt using the same key. The attacker can obtain the key through a passive attack on *A5/2*. The attacker who is in the middle can eavesdrop, change the conversation, perform call theft, etc. The attack applies to all traffic including short message service (SMS) [3].

1.2 Our Contributions

We have designed a new *guess-and-determine* attack on the *A5/1* stream cipher. This attack has a minimum time complexity of approximately 2^{40} and an average complexity of $2^{48.5}$, which is much lesser than a brute-force attack of 2^{64} complexity. For every combination of 19 bits of register R_1, we require storage of 100MB to determine registers R_2 and R_3 for that R_1 input and known keystream KS. This attack has a 100% success rate and requires about 5.65GB storage. With the knowledge of only 11 bits of the known keystream, the attack algorithm is able to determine a set of 64-bit complete state candidates. A complete state candidate is a state candidate with no vacant bits. With every additional clocking round, the number of complete state candidates increases. Thus, the probability of finding the correct state candidate amongst all the complete state candidates increases with every additional round. We provide a detailed description of our new attack along with its implementation and results in Sections 4, 5 and 6.

2 Description of the *A5/1* Stream Cipher

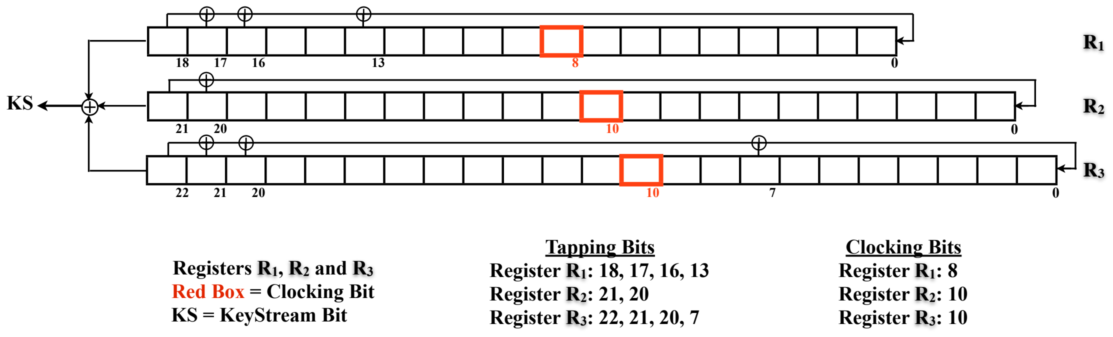


Fig. 1. *A5/1* Stream Cipher

The *A5/1* stream cipher is built from three short linear feedback shift registers (LFSRs) of lengths 19, 22, and 23 bits, denoted 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; the tapping bits of R_2 are at bit positions 20, 21; and the tapping bits of R_3 are at bit positions 7, 20, 21, 22 (Table 1).

Table 1. The *A5/1* Register Parameters

Register	Register Length	Clocking Bit	Primitive Polynimials	Tapping Bits
R_1	19 bits	8	$1+x+x^2+x^5+x^{19}$	18, 17, 16, 13
R_2	22 bits	10	$1+x+x^{22}$	21, 20
R_3	23 bits	10	$1+x+x^2+x^{15}+x^{23}$	22, 21, 20, 7

The tapping bits are pre-determined according to the corresponding primitive polynomials for the registers. A polynomial of degree n over the finite field $GF(2)$ (i.e., with coefficients either 0 or 1) is primitive if it has polynomial order $2^n - 1$. For each register, when the register is clocked, its tapping bits are XORed together and the result is stored in the rightmost bit of the left-shifted register. The three registers are maximal length LFSRs with periods $2^{19} - 1$, $2^{22} - 1$, and $2^{23} - 1$ respectively [14].

The *A5/1* keystream generator works as follows [6]. First, an initialization phase is run. At the start of this phase, all bits of the registers are set to 0. Then the key setup and the *Initialization Vector (IV)* setup are performed. During the initialization phase, all three registers are clocked and the key bits followed by the *IV* bits are XORed with the *most significant bits* (MSBs) of all three registers. Thus, the initialization phase takes an overall of $64 + 22 = 86$ clock-cycles after which initial state S_i is achieved.

Based on this initial state S_i, a warm-up phase is performed where the generator is clocked for 100 clock-cycles and the output is discarded. This results directly in state S_w producing the first output bit 101 clock-cycles after the initialization phase. During the warm-up phase and the stream generation phase that follows, the registers R_1, R_2, and R_3 are clocked irregularly according to the majority function rule [7] depending on the *clocking bits* (CBs) of the three registers. The majority function is a function from n inputs to one output. The value of the operation is true when atleast $\frac{n}{2}$ arguments are true, and false otherwise.

The registers are clocked in a stop/go fashion using the following majority rule: Each register has a single clocking bit (bit 8 for R_1, bit 10 for R_2, and bit 10 for R_3) which decides the clocking pattern for its respective register. In each clock cycle, the majority function of the clocking taps is calculated and only those registers whose CBs agree with the majority function are clocked. At each step either two or three registers are clocked, and that each register has a probability of moving 3 out of 4 times. It is this clocking pattern which makes the stream cipher generate output bits which are random.

During encryption, a total of four cases are possible for clocking pattern of the registers. They are:

Case 1: $CB_1 = CB_2 \neq CB_3$ (Clock R_1 and R_2 only)
Case 2: $CB_1 \neq CB_2 = CB_3$ (Clock R_2 and R_3 only)
Case 3: $CB_1 = CB_3 \neq CB_2$ (Clock R_1 and R_3 only)

Case 4: $CB_1 = CB_2 = CB_3$ (Clock all three registers)

where CB_i denotes the *clocking bit* for register i; $i = (1, 2, 3)$.

After clocking, an output bit is generated from the values of R_1, R_2, and R_3 by XOR-ing their *most significant bits* (MSBs), as shown in Equation 1. This XORed bit is called the *keystream bit* (KS).

$$R_1[18] \oplus R_2[21] \oplus R_3[22] = KS[i], \tag{1}$$

where KS[i] denotes the i^{th} keystream bit, $i = 0$ on initialization and increases by 1 after every clocking round.

After warm-up phase, the *A5/1* produces 228 output bits. For every clock cycle, 114 bits are used to encrypt uplink traffic, while the remaining 114 bits are used to decrypt downlink traffic [9].

3 Known Attacks on the *A5/1*

Here we discuss the known *guess-and-determine* attacks on the *A5/1*. A guess-and-determine attack [16] is a known-plaintext attacks on stream cipher where the attacker guesses some bits of the cipher and the remaining bits are determined from the known keystream bits. The 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 secret key used for encrypting the known plaintext to the known ciphertext. These include Anderson's Attack [1], Golic's Attack [11], Biham-Dunkelman's Attack [4], Keller-Seitz's Attack [13] and Gendrullis-Novotny-Rupp's Attack (also known as the Modified Keller-Seitz Attack) [10]. All these attacks assume to have 64 bits of the keystream (KS) known.

Guess-and-Determine Attacks

The first guess-and-determine attack on the *A5/1* was proposed by Anderson [1]. Anderson suggested to guess 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 biits of R_3 by Equation 1. In the worst-case, each of the 2^{52} determined state candidates need to be verified against the known keystream. This attack was not implemented as Biham-Dunkelman's Attack and Keller-Seitz's Attack had lesser complexity.

Golic proposed an attack that has a complexity of 2^{40} linear equations sets [11]. He suggested 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 is much more complicated since it is based on the solutions of system of linear equations. In practice, this algorithm is not better than the Anderson's approach [1] or Keller-Seitz's [13] approach. In deriving the solution of the system of equations, we additionally require solving 44 linear equations by Gaussian Elimination method. This makes Golic's approach impractical to implement.

Pornin and Stern [17] proposed a *Software-Hardware tradeoff attack* that is based on Golic's approach. But in contrast to Golic's approach, they guess the clocking sequence at the very beginning. The increased assumptions and complexity of the attacks make the actual implementation very difficult and impractical.

The Biham-Dunkelman attack [4] is expected to be a thousand times faster than the Anderson's attack [4] or Keller-Seitz's attack [13]. 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. Hence, a lot of pre-computation space is needed.

The attacker guesses 12 bits (i.e., $R_1[(9,18)] \sim R_1[13]$), $R_2[0]$, $R_3[22]$ 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. The attacker must know exactly the location of the information-leaking event where register R_3 is unclocked for 10 consecutive rounds. Such an event will happen one out of 2^{20} possible cipher states. 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. Also, the probability that such an event, where register R_3 is not clocked for consecutive 10 rounds occurs is close to zero. This attack requires a lot of data and pre-computation space. Hence this attack is not practical for implementation.

Keller and Seitz designed a new attack [13] based on the attack proposed by Anderson. But unlike Anderson, 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* [6] 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 by guessing the clocking bit of R_3 such that R_3 will not be clocked. Hence, all states arising out of the contradictory guess neither need to be computed further on nor checked afterwards. The authors further reduce the complexity by not only discarding the incorrect possibilities for $R_3[22]$ in case of contradiction, but also limit the number of choices to the one of not-clocking R_3, if this is possible without any contradiction. If a case arises where $R_1[8] = R_2[10]$ and $R_3[10]$ has to be guessed, then 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 [10] proposed a modification to the Keller-Seitz attack. Unlike Keller-Seitz [13], the authors only discard the wrong possibilities for the clocking bit of register R_3 that would lead to a contradiction. But if no contradiction exists, they check all possibilities of the clocking bit of R_3, which means the case of clocking and not-clocking R_3. Thus, every possible state candidate is taken into account, hence giving us a success probability of 100%.

Besides Golic [11] and Babbage [2], Biryukov-Shamir-Wagner [5] proposed an attack with a complexity of 2^{48} requiring about 300GB storage, where the online phase of the attack can be executed within minutes with a 60% success probability.

Barkan-Biham-Keller [3] also proposed another attack along these lines. However, in the precomputation phase of such an attack huge amounts 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 making actual implementations of such attacks very difficult.

4 A New Attack on the *A5/1*

Our approach is based on the *guess-and-determine attack* proposed by Anderson [1], but with several modifications. With 64 bits of keystream (KS) known, all bits of register R_1 are guessed (known) and all bits of registers R_2 and R_3 are determined. But unlike the approaches of Anderson [1], Golic [11], Biham-Dunkelman [4] and Keller-Seitz [13], we consider all possible cases i.e., no case is discarded. At the end, we come up with about $2^{48.5}$ possible state candidates, which is much smaller than the exhaustive search where we have 2^{64} state candidates (refer Section 4.4). Hence this attack is better than the exhaustive search approach. The minimum time complexity (lower bound) for the attack is approximately $2^{45.2}$, which is attained after 11 rounds of clocking.

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

The *determination phase* generates all possible state candidates after the *warm-up phase* [6] is completed. Let t_2 and t_3 denote the number of times the registers R_2 and R_3 are clocked, respectively. Everytime a register is clocked, increase the counter of that register by one. Initialize the algorithm by giving the input of known keystream bits (KS) and guessing all bits of the smallest register R_1 (Figure 2).

Processing-Phase1 Compute the most significant bits (MSBs) 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 above equation. But if $R_2[21]$ and $R_3[22]$ are unknown, then there exist four possible combinations for the unknown bits; i.e. 00, 01, 10 and 11. But the above equation reduced the number of possibilities to two. The two possibile combinations that satisfy the equation are:

– *If* $R_1[18] = KS[i]$, *then* $R_2[21] = R_3[22] = 0$ or $R_2[21] = R_3[22] = 1$.
– *If* $R_1[18] \neq KS[i]$, *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 half. During initialization, i is set to 0, and with every additional clocking round, the value of i increases by 1. Note that i also denotes the total number of clocking rounds that have taken place.

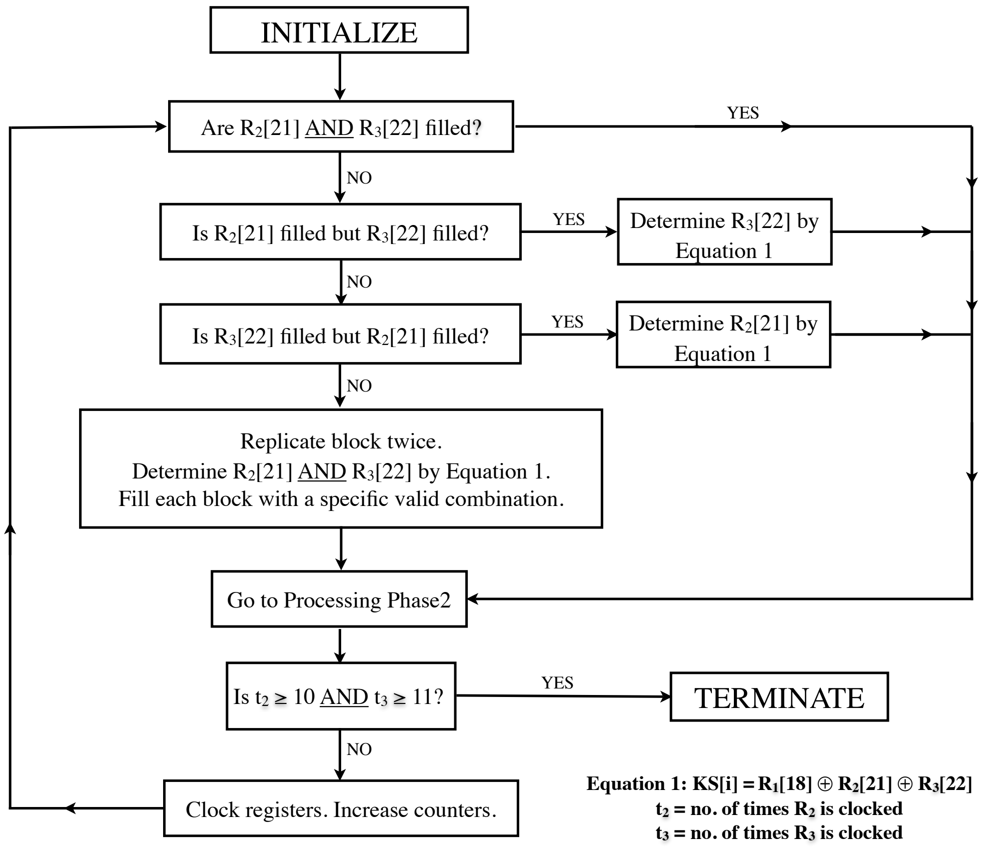

INITIALIZE
Are R2[21] AND R3[22] filled?
YES
NO
Is R2[21] filled but R3[22] filled?
YES
Determine R3[22] by Equation 1
NO
Is R3[22] filled but R2[21] filled?
YES
Determine R2[21] by Equation 1
NO
Replicate block twice.
Determine R2[21] AND R3[22] by Equation 1.
Fill each block with a specific valid combination.
Go to Processing Phase2
Is t2 ≥ 10 AND t3 ≥ 11?
YES
TERMINATE
NO
Clock registers. Increase counters.
Equation 1: KS[i] = R1[18] ⊕ R2[21] ⊕ R3[22]
t2 = no. of times R2 is clocked
t3 = no. of times R3 is clocked

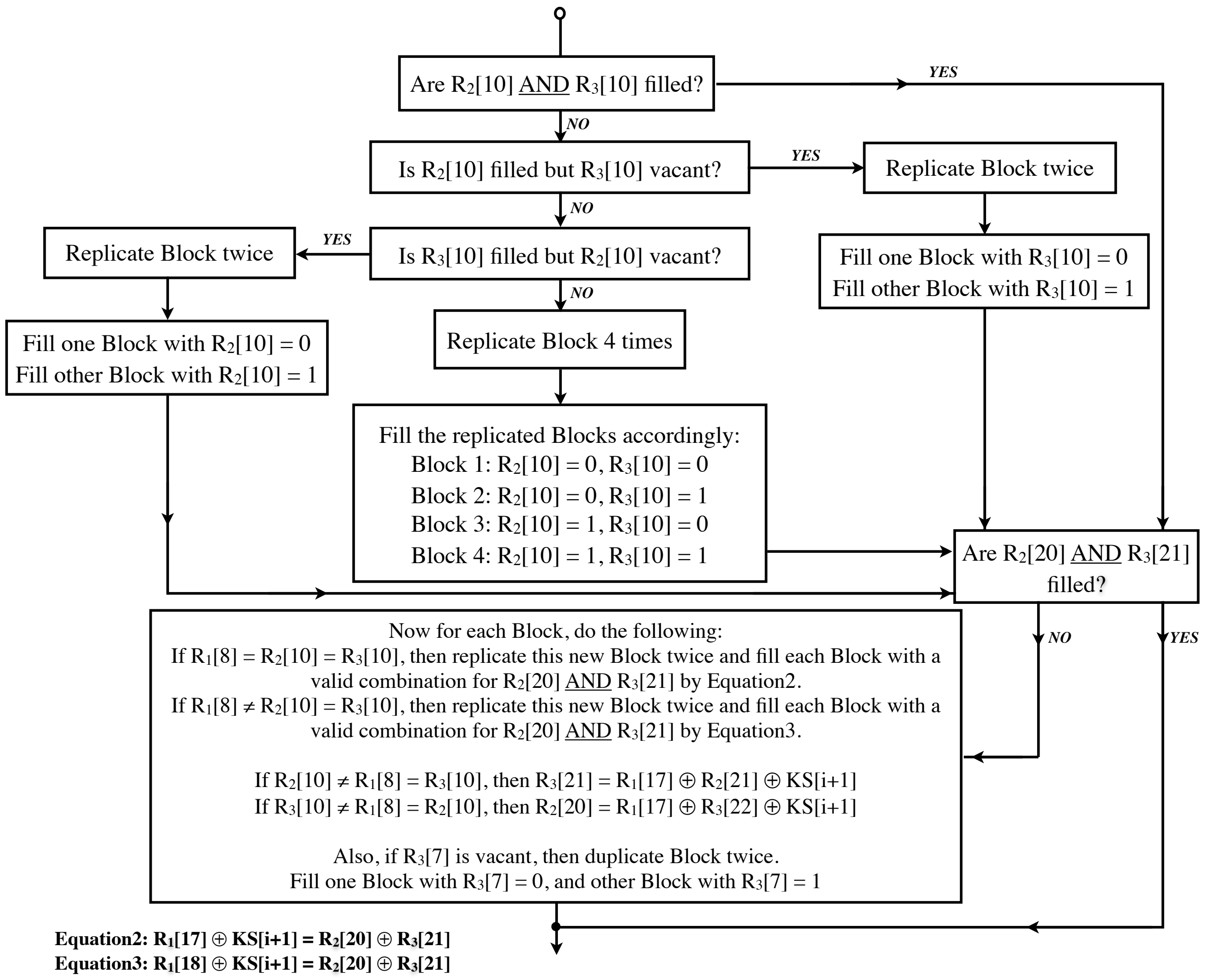
Fig. 2. Determination Phase of the Attack (Processing-Phase1)

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 $R_2[10] = 1$, $R_3[10] = 1$.

Thus, all possible combinations are taken into consideration (Figure 3).

Now consider the bits $R_2[20]$ and $R_3[21]$. 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



Are R2[10] AND R3[10] filled?
YES
NO
Is R2[10] filled but R3[10] vacant?
YES
Replicate Block twice
NO
Is R3[10] filled but R2[10] vacant?
YES
Replicate Block twice
Fill one Block with R3[10] = 0
Fill other Block with R3[10] = 1
NO
Fill one Block with R2[10] = 0
Fill other Block with R2[10] = 1
Replicate Block 4 times
Fill the replicated Blocks accordingly:
Block 1: R2[10] = 0, R3[10] = 0
Block 2: R2[10] = 0, R3[10] = 1
Block 3: R2[10] = 1, R3[10] = 0
Block 4: R2[10] = 1, R3[10] = 1
Are R2[20] AND R3[21] filled?
NO
YES
Now for each Block, do the following:
If R1[8] = R2[10] = R3[10], then replicate this new Block twice and fill each Block with a valid combination for R2[20] AND R3[21] by Equation2.
If R1[8] ≠ R2[10] = R3[10], then replicate this new Block twice and fill each Block with a valid combination for R2[20] AND R3[21] by Equation3.
If R2[10] ≠ R1[8] = R3[10], then R3[21] = R1[17] ⊕ R2[21] ⊕ KS[i+1]
If R3[10] ≠ R1[8] = R2[10], then R2[20] = R1[17] ⊕ R3[22] ⊕ KS[i+1]
Also, if R3[7] is vacant, then duplicate Block twice.
Fill one Block with R3[7] = 0, and other Block with R3[7] = 1
Equation2: R1[17] ⊕ KS[i+1] = R2[20] ⊕ R3[21]
Equation3: R1[18] ⊕ KS[i+1] = R2[20] ⊕ R3[21]

Fig. 3. Determination Phase of the Attack (Processing-Phase2)

bits are vacant, there are four possible combinations for these bits; i.e., 00, 01, 10 and 11. But Equation 2 and Equation 3 reduce them to two possibilities. This reduces the number of possible cases by half i.e., 50% save.

If only one of these bits is vacant, there are two possibilities for the vacant bit i.e., 0 or 1. But this is reduced to only one possibility (Figure 3). 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 till $t_2 < 10$ and $t_3 < 11$. Once 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 atleast 10 times and register R_3 has to be clocked atleast 11 times to determine all the bits of that register.

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 correct state candidate (i.e., 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 to obtain a set of complete state candidates is 11. The number of possible state candidates increases with every additional round. Hence, the probabilitity of finding the correct state candidate 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 till a contradiction of bit-wise matching occurs. If all the output bits match with the given 64 KS bits and no contradiction occurs, then that state candidate is the correct state candidate. Hence, we have found the correct state candidate amongst all the possible state candidates obtained.

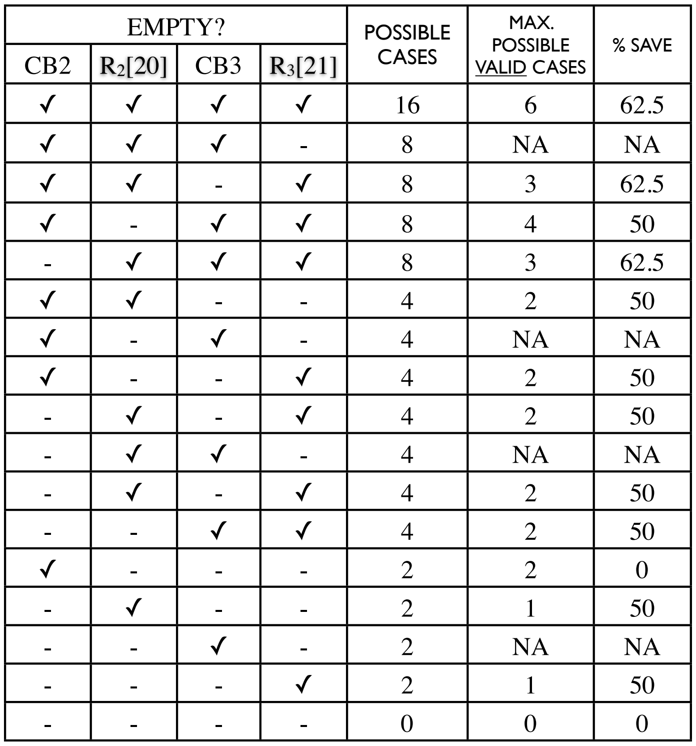
5 Analysis of the Attack

We now discuss each phase of the attack step-by-step. After initialization, we perform the first step of implementation, i.e., the *determination phase*. The state candidate has all bits of register R_1 known and all other bits vacant. According to protocol, the determination phase determines the most significant bits (MSBs) of registers R_2 and R_3 (i.e., $R_2[21]$ and $R_3[22]$) by processing-phase1; the clocking bits of R_2 and R_3 (i.e., $R_2[10]$ and $R_3[10]$), bit $R_3[7]$ and if possible, bits $R_2[20]$ and $R_3[21]$ by processing-phase2.

Now we consider the first stage of the determination phase i.e., processing-phase1. The MSBs of registers R_2 and R_3 have to be determined. The number of possible combinations reduces from four to two by Equation 1. Thus saving two combinations, i.e., a saving of 50%. During the implementation of further rounds, there is a possibility where only one of the MSBs of R_2 or R_3 is vacant. We determine these vacant bit(s) by Equation 1.

We now proceed to processing-phase2 of the determination phase. Here we first consider the four vacant bits: $R_2[10]$ (CB of R_2), $R_3[10]$ (CB of R_3), $R_2[20]$ and $R_3[21]$. But all these four bits (except the first step after initialization) may not be vacant together at all times. In the following table (Figure 4), we consider all possible cases of these four bits being empty, and the number of maximum possible valid combinations that exist as a result of Equation

1. We now consider the bit $R_3[7]$. There are two possibilities for this bit, i.e., 0 and 1. But we cannot eliminate any case by any method. Hence, we need to consider both cases.



EMPTY?				POSSIBLE CASES	MAX. POSSIBLE VALID CASES	% SAVE
CB2	R2[20]	CB3	R3[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

Fig. 4. All possibilities during Processing-Phase2

Whenever the CB of register R_3 is vacant, the bit $R_3[21]$ has to be vacant too. Hence there are some cases in the following table which are *not applicable* (NA). The last column depicts the percentage of the total possible cases that are discarded due to the attack algorithms.

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 our algorithms give only 24 valid possible combinations. Thus saving 104 combinations i.e., 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 (till round 10) can be approximated by the formula $12 * 5^{n-1}$, where n denotes the n^{th} round, $n \in \mathbb{Z}^+$, $n < 11$. It is only after the 11^{th} round that we will get the first set of complete state candidates (with all registers full). When 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

In this section, we discuss in detail a probabilistic approach to determine the time complexity, the storage requirement and success probability of the 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 correct state candidate (key) is 15.5 and the average number of complete state candidates obtained after 15.5 rounds is $2^{48.5}$.

6.1 Time complexity

Now we understand the exact time-complexity of the attack algorithm. According to the algorithm, work done is during:

- Checking
- Replication
- Filling vacant bits

It is reasonable to assume that checking and filling takes negligible amount of time. Hence, we can safely assume that the unit of our time complexity measurement should be the number of replications needed for the algorithm to terminate, where one unit 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 $2^{48.5}$ complete state candidiates, i.e., $2^{48.5}$ replications.

The number of bits between the clocking bit (CB) and the most significant bit (MSB) 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 determine all the bits of that register is atleast 10 and 11 respectively. The minimum number of KS bits required to obtain a set of complete state candidates (with no vacant bits) 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.

With every clocking round, the number of complete state candidates increases. Hence, the probability of finding the correct state candidate increases with every round of clocking.

According to the *majority function* of the *clocking rule* for the *A5/1*, a register will get clocked 3 out of 4 times. At every clocking cycle, atleast two registers will get clocked. Let n_1 be the event that registers R_2 and R_3 are clocked together, and n_2 be the event that register R_1 is clocked either with register R_2 or with register R_3. The probabilities that events n_1 and n_2 occur are given by $P(n_1)$ and $P(n_2)$ respectively. Let n_2' be the event that only registers R_1 and R_2 are clocked. Let n_2'' be the event that only registers R_1 and R_3 are clocked. The probabilities that events n_2' and n_2'' occur are given by $P(n_2')$ and $P(n_2'')$ respectively. Hence, one can conclude that $P(n_1) = P(n_2) = \frac{1}{2}$. Thus,

$$P(n_1) + P(n_2) = \frac{1}{2} + \frac{1}{2} = 1$$

and

$$P(n_2) = P(n_2') + P(n_2'') = \frac{1}{4} + \frac{1}{4}.$$

i.e.,

$$P(n_1) + P(n_2') + P(n_2'') = 1$$

Registers R_2 and R_3 have to be clocked atleast 10 and 11 times respectively to determine all bits of that register i.e., to obtain a set of complete state candidates. They may be clocked by n_1 or n_2.

Let X be the random variable denoting the number of clocking cycles needed to obtain complete state candidate. Let x_1 be the number of clocking cycles needed for event n_1, x_2 for n_2' and x_3 for n_2''. Here, $x_1 = 10$, $x_2 = 10$ and $x_3 = 11$. Then the expectation for this variable X is defined as

$$E[X] = \frac{x_1 * P(n_1) + x_2 * P(n_2') + x_3 * P(n_2'')}{P(n_1) + P(n_2') + P(n_2'')}$$

$$\Longrightarrow \frac{10 * \frac{1}{2} + 2 * (10 * \frac{1}{4} + 11 * \frac{1}{4})}{\frac{1}{2} + \frac{1}{4} + \frac{1}{4}} = 15.5$$

An Experiment We implement normal encryption of *A5/1* using random inputs for all three registers. The aim of this experiment is to determine the average number of clocking rounds needed for register R_2 and R_3 to be clocked atleast 10 and 11 times respectively. We performed this experiment thrice, each time with 250 inputs. The average number of clocking rounds needed turned out to be 15.51 with a standard deviation of 1.785. Hence, the experimental results corroborate with the theoretical proof.

The number of clocking rounds necessary to obtain a set of complete state candidates is approximately 15.5. This set of complete state candidates would contain the correct state candidate (key) with a high probability.

6.2 Storage Requirement and Success Probability

As discussed in Section 4.1, the minimum number of KS bits required to generate a set of complete state candidates (all bits filled) is 11. With every additional clocking round, the number of complete state candidates increase. We can perform the post-processing-phase of the attack after every round simultaneously while performing the determination phase of the next round. Hence, the probability of finding the correct state candidate also increases with every round. But we require atleast 64 KS bits for the post-processing-phase of the attack to bit-wise match and check for the correct state candidate.

The number of possible state candidates obtained after the 11^{th} round is approximately $2^{45.2}$. The total number of complete state candidates is approximately $2^{39.2}$ i.e., around 1.6% of total number of possible state candidates obtained after the 11^{th} round. Thus, the minimum complexity of the attack (lower bound) is around 2^{40}. As stated earlier, the number of possible state candidates increases with every round. Here we plot a table (Table 2) of the results of the experimental data. 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 percentage of number of complete state candidates over the total number of state candidates for that round. All values of the experimental data in the table are approximated to one decimal place.

Table 2. Storage Requirement and Success Probability

No. of Rounds	Total State Candidates	Complete State Candidates	$\frac{Complete}{Total} * 100$
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: In each round, the number of possible choices reduce to atleast half in each case (refer Figure 4). Hence, a minimum saving of 50% takes place in every round. As stated in Section 6.1, the average number of rounds to get the correct state candidate (key) is around 15.5. In each round, we save atleast half the possible cases. Hence for 15.5 rounds we save $(\frac{1}{2})^{15.5}$ cases. Thus, the average number of complete state candidates obtained after 15.5 rounds will be $(2^{64})(\frac{1}{2})^{15.5} = 2^{48.5}$. The correct state candidate (key) would be amongst the set of complete state candidates obtained after the 15th round, with a probability of 50%.

Storage For each guess of 19 bits of register R_1, we require storage of 100MB to determine registers R_2 and R_3 for that R_1 input and known keystream KS. If we do not find the correct state candidate for this combination of 19 bits of register R_1, then we discard this guess and make a new guess for the bits of register R_1. In the worst-case, we need to check all 2^{19} possible input combinations of register R_1. The attack has a 100% success probability and requires about 5.65GB storage.

7 Conclusion

Our attack is based on the *guess-and-determine* approach proposed by Anderson [1], but with several modifications. In this attack, all bits of the first register R_1 are guessed (known) and all bits of registers R_2 and R_3 are determined with 64 bits of the known *keystream* (KS).

This attack has an average time complexity of $2^{48.5}$, which is much smaller than the exhaustive search where the time complexity is 2^{64}. Hence this attack is better than the exhaustive search approach. The minimum time complexity (lower bound) for the attack is approximately 2^{40}, which is attained after 11 rounds of clocking. The average number of rounds necessary to obtain the correct key from the set of complete state candidates is 15.5. The probability of success in finding the key after 15 rounds, by post-processing phase, is more than 50%. With every round of clocking after 11 rounds, the number of complete state candidates increases. Thus, the probability of finding the correct state candidate increases with every clocking round. The attack is successful with 100% probability and requires aboubt 5.65GB storage.

References

1. Anderson, R., *A5* (was: Hacking digital phones), http://yarchive.net/phone/gsmcipher.html, Newsgroup Communication, 1994.
2. Babbage, S., *A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers*, European Convention on Security and Detection, 1995.
3. Barkan, E., Biham, E., and Keller, N., *Instant Ciphertext-only Cryptanalysis of GSM Encrypted Communication*, Technical Report CS-2006-07, Technion, 2006.
4. Biham, E. and Dunkelman, O., *Cryptanalysis of the A5/1 GSM Stream Cipher*, In Proc. of Indocrypt'00, vol 1977 of LNCS. Springer-Verlag, 2000.
5. Biryukov, A., Shamir, A. and Wagner, D., *Real Time Cryptanalysis of A5/1 on a PC*, In Proc. of FSE'00, vol 1978 of LNCS, pp 1-18. Springer-Verlag, 2001.
6. Briceno, M., Goldberg, I., and Wagner, D., *A Pedagogical Implementation of the GSM A5/1 and A5/2 "voice privacy" Encryption Algorithms*, http://cryptome.org/gsm-a512.html, 1999.
7. Donald, E., *Introduction to Combinatorial Algorithms and Boolean functions. The Art of Computer Programming. 4.0*, Upper Saddle River, NJ: Addison-Wesley. pp 64-74, 2008.
8. Ekdahl, P. and Johansson, T., *Another Attack on A5/1*, IEEE Transactions on Information Theory, 49(1), pp 284-289, 2003.
9. Gendrullis, T., *Hardware-Based Cryptanalysis of the GSM A5/1 Encryption Algorithm*, Diploma thesis, Ruhr-University Bochum 2008.
10. Gendrullis, T., Novotny, M., and Rupp, A., *A Real-World Attack Breaking A5/1 within Hours*, Proc. of CHES'08, vol 5154 of LNCS, pp 266-282. Springer-Verlag, 2008.
11. Golic, J., *Cryptanalysis of Alleged A5 Stream Cipher*, In Proc. of Eurocrypt'97, vol 1233 of LNCS, pp 239-255. Springer-Verlag, 1997.
12. Hillebrand, Friedhelm, *GSM and UMTS: The creation of Global Mobile Communication*, Wiley 2002.
13. Keller, J., and Seitz, B., *A Hardware-Based Attack on the A5/1 Stream Cipher*, http://pv.fernuni-hagen.de/docs/apc2001-final.pdf, 2001.
14. Kostopoulos, G., Sklavos, N., Galanis, M., and Koufopavlou, O., *VLSI Implementation of GSM Security: A5/1 and W7 Ciphers*, In Proc. of IEEE Workshop on Wireless Circuits and Systems (IEEE WoWCAS'04), Canada, 2004.
15. Maximov, A., Johnasson, T. and Babbage, S., *An Improved Correlation Attack on A5/1*, In Proc. of SAC'04, vol 3357 of LNCS, pp 239-255. Springer-Verlag, 2005.
16. Menezes, A., van Oorschot, P., and Vanstone, S., *Handbook of Applied Cryptography*, CRC Press, 1997.
17. Pornin, T., and Stern, J., *Software-hardware Trade-offs: Application to A5/1 Cryptanalysis*, Proc. of CHES'00, vol 1965 of LNCS, pp 318-327, Springer-Verlag, 2000.