Hindawi
Complexity
Volume 2023, Article ID 7249127, 9 pages
https://doi.org/10.1155/2023/7249127

Research Article

A New Guess-and-Determine Method for Cryptanalysis of the GSM Encryption

Ashish Jain,[1] **Inderjeet Kaur**,[2] **Akhilesh Kumar Sharma**,[1] **Nirmal Kumar Gupta**,[1] **and Partha Chakraborty**[3]

[1]*Department of Information Technology, Manipal University Jaipur, Jaipur, India*
[2]*Department of Computer Science and Engineering, Ajay Kumar Garg Engineering College, Ghaziabad, India*
[3]*Department of Computer Science and Engineering, Comilla University, Comilla, Bangladesh*

Correspondence should be addressed to Partha Chakraborty; partha.chak@cou.ac.bd

Received 8 September 2022; Revised 1 December 2022; Accepted 7 January 2023; Published 18 February 2023

Academic Editor: Rosa M. Lopez Gutierrez

Cryptanalysis is the process of finding flaws or oversights in an encryption algorithm. Nearly, all encryption algorithms are carefully examined through cryptanalysis to determine the security of the system in which the encryption algorithm has been employed. A5/1 is a well-known encryption algorithm which is inbuilt in mobile phone for securing GSM communication, and therefore, cryptanalysis of this algorithm is also important. A5/1 consists of three linear feedback registers of lengths 23, 22, and 19 bits. Due to the nonlinear clocking mechanism of A5/1, cryptanalytic attacks of guess-and-determine (GD) nature are efficient and more successful. In this paper, we propose a new low keystream GD attack on GSM encryption algorithm A5/1. The basic idea of GD attack is guessing some portion of the registers of A5/1 and determining remaining portion of the registers via the relationship between the register's state and the known intercepted keystream. The guessed and determined register's state is validated by running the cipher forward from that state. If the intercepted keystream matches the generated keystream, we accept it. Otherwise, we discard and try the attack again. The computational complexity and the success rate of the proposed attack are O (2^{52}) and 96.6%, respectively.

1. Introduction

There are two categories of encryption algorithm (or cipher), namely, symmetric cipher and asymmetric cipher. Symmetric ciphers are further classified as stream cipher and block cipher. A stream cipher is an approach of encrypting data one bit at a time using an encryption algorithm and a secret key. As opposed to a stream cipher, block cipher encrypts data simultaneously in fixed-size blocks.

Cryptanalysis is the process of finding flaws or oversights in an encryption algorithm. Nearly, all encryption algorithms are carefully examined through cryptanalysis to determine the security of the system. A5/1 is a well-known synchronous stream cipher which is implemented on the hardware of GSM mobile phones for providing over-the-air communication privacy [1–5], and therefore, cryptanalysis of A5/1 stream cipher is also important.

For the construction of synchronous stream ciphers, linear feedback shift registers (LFSRs) are often used because these registers can be easily implemented in hardware, have good statistical properties, and can produce keystream bits at or near the clock speed. Historically, stream ciphers were designed to be employed in real-time systems because of their high-speed encryption capability. However, modern block ciphers are also capable of encrypting at high data rates, e.g., AES, PRESENT [6], and 3DES [7]. Nevertheless, some important applications where the computational resources are limited still prefer stream ciphers to encrypt large quantities of fast streaming data [8]. Due to such important applications, security analysis of stream ciphers is very important.

Nearly, all cryptographic and security algorithms undergo security analysis process and are carefully examined to establish the practical security of the system. In other words, the goal of the cryptanalyst is to find flaws or insecurities in a cryptography system so that the system can be upgraded. Typically guess-and-determine (GD) attacks, time-memory-data trade-off (TMDTO) attacks, correlation attacks, and statistical distinguishing attacks are few methods to cryptanalyze stream ciphers. In 1994, when the approximate design of A5/1 was leaked, it was cryptanalyzed by Anderson [1], Babbage [9], and Golić [10]. The exact design of A5/1 was revealed when it was reverse engineered by Briceno et al. [11] from GSM telephone [12]. In the literature, several attacks have been studied on the "exact design of A5/1" (for example, [4, 5, 13–22]). However, previous proposed attacks (except [16, 21]) are of interest from the theoretical aspects because of their deficiency in practicality due to needs of large amount of keystream, lot of precomputation, high computational demands, huge storage requirements, and/or deficiency in considering asynchronous (or irregular) clocking of A5/1. It should be noted that irregular clocking of A5/1 is very important. If irregular clocking of A5/1 is not considered, then the randomness characteristics of the A5/1 generator weakens. Consequently, cryptanalysis of A5/1 becomes simple and can be performed in less computation. It is worth pointing that due to the irregular clocking of A5/1, GD attacks are often mounted on it. Moreover, such types of attacks are common and are very successful against stream ciphers. The basic idea of such type of attack is guessing some portion of the internal state and determining remaining portion via the relationship between the internal state and the known intercepted keystream. The guessed and determined internal state is validated by running the cipher forward from that state. If the intercepted keystream matches the generated keystream, we accept it. Otherwise, we discard the current candidate and try the attack again [22–24]. In this work, we present a new low keystream GD attack of A5/1 and compare our attack with previously proposed GD and TMDTO class of attacks. The remainder of the work is organized as follows. In Section 2, we present the design of A5/1. In Section 3, we present a new low keystream attack on the GSM encryption algorithm A5/1. In Section 4, a review on the previous A5/1 attacks is presented followed by conclusion in Section 5.

2. Description of A5/1: A GSM Stream Cipher

For GSM conversation between two communication parties, e.g., A and B, a sequence of the frame is transmitted. Each frame is transmitted in 4.615 milliseconds that contains 228 bits, where 114 bits represent the communication from A to B and the remaining 114 bits represent return communication. Each frame also contains a frame counter F_n of 22 bits which is publicly known. For each GSM conversation, a new 64-bit session key k is generated. The session key followed by a frame counter is used to set the initial state of A5/1 [4, 5]. Three LFSRs *REG*1, *REG*2, and *REG*3 of lengths 19, 22, and 23 bits, respectively, are used to design A5/1 keystream generator (see Figure 1).

Figure 1 shows that each register has a clocking tap at 8th position in *REG*1, at 10th position in *REG*2, and at 10th position in *REG*3. Each register also has a primitive feedback polynomial (see Section 3, Paragraph 1, for details). All LFSRs are clocked in a stop/go fashion according to the majority rule, i.e., the registers that have the same clocking bit must be clocked simultaneously. It should be noted that at each clock, either two or three registers are clocked, i.e., each register moves with probability 3/4 and stops with probability 1/4.

A5/1 operates as follows: the first step is an initialization step in which all LFSRs are set to "0" and then all are clocked 64 times linearly, and in parallel, successive bits of the session key are XORed to the feedback of each register. In the second step, all LFSRs are clocked 22 times (again linearly), and in parallel, consecutive bits of the frame counter are XORed to the feedback of each register. In this way, at the end of 86th $(64 + 22)$ clock cycle, an initial state S_i is obtained. In other words, a linear operation for 86 clock cycles is performed for loading key and frame on the A5/1 keystream generator. Finally, on the use of initial state S_i, A5/1 carries out the warm-up phase. In this phase, all LFSRs are clocked in a nonlinear way (according to the majority rule) for 100 clock cycles and the output is discarded. Afterward, A5/1 is clocked nonlinearly for 228 clock cycles that produce 228-bit keystream. The keystream combines with 228 bits of the plaintext, and in this way, the ciphertext of 228 bits is generated. For more details about A5/1, the readers can refer [4, 5, 15].

3. Proposed Attack

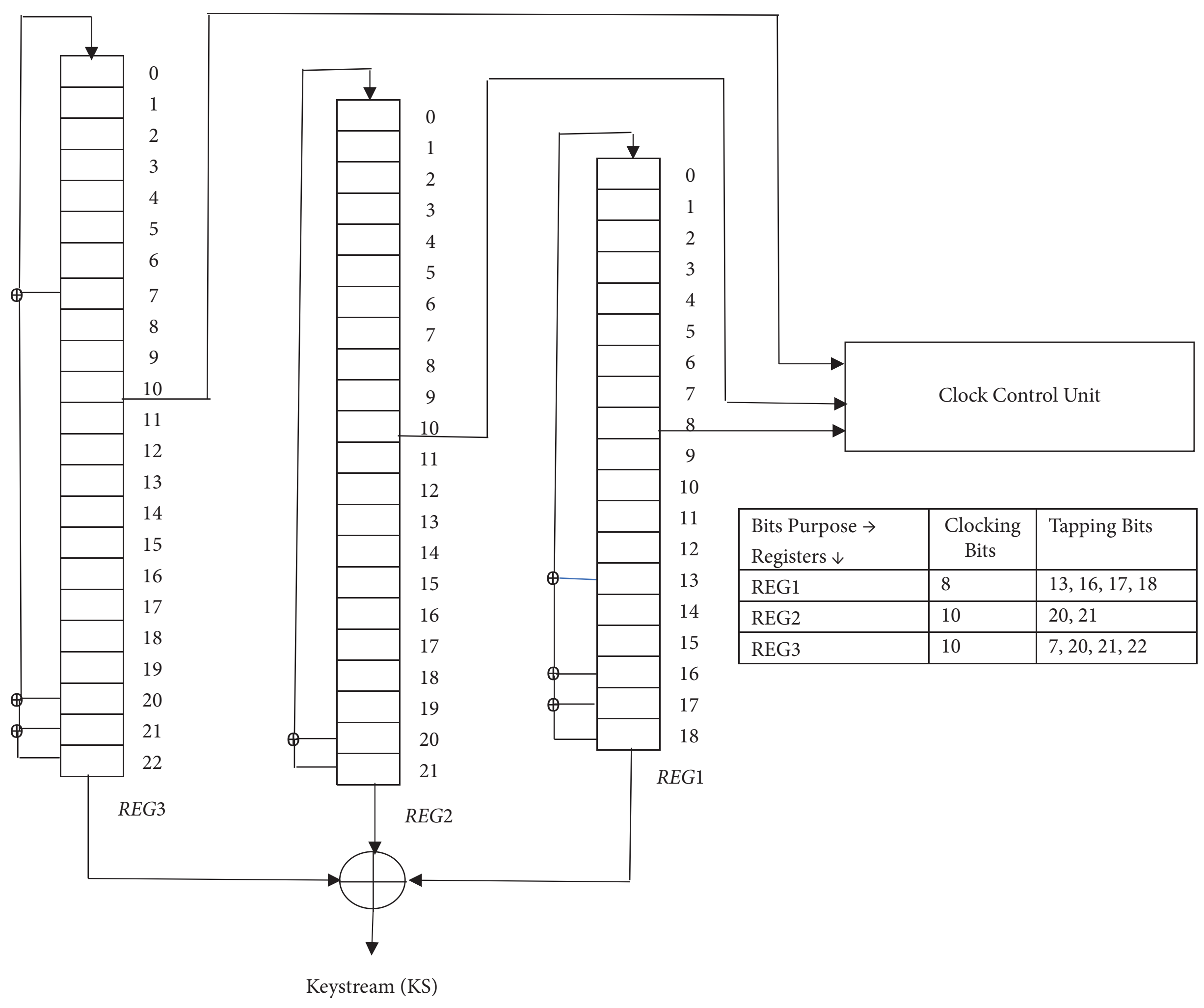
Figure 1 shows that tapping bits, i.e., {13, 16, 17, 18}, {20, 21}, and {7, 20, 21, 22}, are involved in primitive polynomials of *REG*1, *REG*2, and *REG*3, respectively. Whenever a register is clocked, its output is feedback at the least significant bit (LSB) position of the respective register. However, before storing the output at the LSB, bits below the LSB position are shifted downwards, i.e., REG$i[j]$←REG$i[j$–1] (i indicates the register number and $j > 0$). In our method of attack, we guess all bits of *REG*1 and then determine the contents of *REG*2 and *REG*3 using 64 known keystream (KES) bits. The attacks that use very small amount of KES bits, e.g., 64 bits, are called low keystream attacks. In contrast to previously known low keystream attacks on the exact design of A5/1, computational complexity of the proposed attack is 2^{52} which is comparatively better. Also, the rate of success is 96.6% which is remarkable.

3.1. Determination Phase

3.1. Determination Phase. This phase determines the initial internal state for all possible state candidates. Since *REG*1 is guessed, we determine *REG*2 and *REG*3 which are unknown. Firstly, we compute the MSB of *REG*2 and *REG*3 by substituting MSB of *REG*1 and the first KES bit in the following equation:

$$\mathrm{REG2}[21] \oplus \mathrm{REG3}[22] = \mathrm{REG1}[18] \oplus \mathrm{KES}[0]. \tag{1}$$

Initially, MSB of *REG*2 and *REG*3 is unknown, and obviously four possibilities exist together. However, we reduce four possibilities to two possibilities by applying following two conditions:



Bits Purpose → Registers ↓	Clocking Bits	Tapping Bits
REG1	8	13, 16, 17, 18
REG2	10	20, 21
REG3	10	7, 20, 21, 22

Figure 1: A5/1 stream cipher.

$$\begin{aligned}&\{1\}\text{If REG1}[18] = \text{KES}[0]\text{then REG2}[21] = \text{REG3}[22] = 0\,(\text{or})\text{REG2}[21] = \text{REG3}[22] = 1,\\&\{2\}\text{If REG1}[18] \neq \text{KES}[0]\text{then REG2}[21] = 0, \text{REG3}[22] = 1\,(\text{or})\text{REG2}[21] = 1, \text{REG3}[22] = 0.\end{aligned} \quad (2)$$

The above condition {1} or {2} reduces the amount of possible state candidates to half. Afterward, by replication (for example, [25, 26]) of possible state candidates, we determine *REG*2 and *REG*3. For this purpose, first we concentrate on the clocking bit of *REG*2 and *REG*3 registers. Since we need to guess all possible combinations of clocking bit of *REG*2 and *REG*3, there are three possibilities that are as follows:

(1) If *REG*2[10] and *REG*3[10] are unknown, then the state candidate is replicated as follows: load first copy by *REG*2[10] = 0, *REG*3[10] = 0; second copy by *REG*2[10] = 0, *REG*3[10] = 1; third copy by *REG*2 [10] = 1, *REG*3[10] = 0; and finally fourth copy by *REG*2[10] = 1, *REG*3[10] = 1.

(2) If *REG*2[10] is known and *REG*3[10] is unknown, then state candidate is replicated as follows: load first copy by *REG*3[10] = 0 and second copy by *REG*3 [10] = 1.

(3) If *REG*2[10] is unknown and *REG*3[10] is known, then the state candidate is replicated as follows: load first copy by *REG*2[10] = 0 and second copy by *REG*2 [10] = 1.

We consider the second most significant bit of *REG*2 and *REG*3, i.e., *REG*2[20] and *REG*3[21]. If *REG*2 and *REG*3 are clocked together, then *REG*2[20] and *REG*3[21] are shifted downwards and turn out to be new MSB (all possibilities of clocking and un-clocking of *REG*2 and *REG*3 are described in Algorithm 1). There are four possibilities for these new bits, but equations (3) and (4) reduce them in two possibilities. That is, equations (3) and (4) reduce the amount of possible state candidates to half. Moreover, as we are considering the initialization of tapping bit *REG*3[20], in further rounds of attack when *REG*3 is clocked, *REG*3[21] remains known and *REG*3[20] becomes unknown, since downwards shifting *REG*3[21]←*REG*3[20].

$$\text{REG2}[20] = \text{REG3}[21] \oplus \text{REG1}[17] \oplus \text{KES}[i+1], \quad (3)$$

$$\text{REG2}[20] = \text{REG3}[21] \oplus \text{REG1}[18] \oplus \text{KES}[i+1]. \quad (4)$$

Consider the remaining tapping bits of *REG*3, i.e., *REG*3 [20] and *REG*3[7]. At the time of initialization, for each bit position, we make two replicas and each one is initialized by both possibilities (0 and 1). Now, we are in the situation of finding unknown bits of *REG*2 and *REG*3 by nonlinear clocking of A5/1. In every clocking of *REG*2, two bits become known: one in the lower part of the clocked register and another in the upper part of the clocked register, which implies that two bits become unknown. However, in the case of *REG*3, during first three rounds of clocking, three bits become known and three bits become unknown. For instance, *REG*3 is clocked just after initialization of *REG*1. As a result, bits of register *REG*3 are shifted downwards, i.e., *REG*3[22], *REG*3[21], *REG*3[11], and *REG*3[8] are assigned known value of *REG*3[21], *REG*3[20], *REG*3[10], and *REG*3 [7], respectively, and the XORing result of feedback polynomial is set to *REG*3[0]. In this way, *REG*3[0], *REG*3[8], and *REG*3[11] become known and *REG*3[7], *REG*3[10], and *REG*3[20] become unknown and the remaining positions of *REG*3 remain unknown since downwards shifting is done by an unknown value. Here, the counter associated with *REG*3 is increased by 1. Afterward, *REG*3[7], *REG*3[10], and *REG*3 [20] are guessed again for executing further rounds of attack, but after three rounds of clocking, we need not to guess *REG*3[10] because the number of bits after the 7th index is three (i.e., 8th, 9th, and 10th). A complete iteration-based determination algorithm for determining contents of *REG*2 and *REG*3 is shown in Algorithm 1.

Algorithm 1, which is the determination algorithm, determines all bits of *REG*2 and *REG*3 by utilizing input, i.e., consecutive 64 bits of KES and guessing bits of *REG*1. Let *REG*2 and *REG*3 be clocked c_2 and c_3 times, respectively. Whenever registers are clocked, we update the associated counter by one. Since the partial length between clocking tap and MSB of *REG*2 is "10" as well as between clocking tap and second MSB of *REG*3 is "10", Algorithm 1 is repeated till c_2 and c_3 reach 10. In this way, both registers (i.e., *REG*2 and *REG*3) need to be clocked at least ten times. At this moment, *REG*2 and *REG*3 are completely determined. We observed that for obtaining a reasonable number of complete state candidates (a state candidate with all bits is known can be referred to as a complete state candidate), minimum 10 *KES* bits are required, which is only possible when both registers *REG*2 and *REG*3 are clocked together for 10 consecutive rounds. An important question is that in how many rounds, counters c_2 and c_3 will reach 10, which is the highest probability (about 90% to 100%) of determining the key. We will see the answer to this question in Section 3.3.2.

3.2. Processing Phase. This phase of the attack searches for the key by verifying complete state candidates. In the verification process, we encrypt each complete state candidate by A5/1 and match the generated output bit with the corresponding known *KES* bit. This process (i.e., clocking and bit-wise matching) continues till a contradiction occurs. When no contradiction occurs, it implies that the cipher's internal state after the warm-up phase has been obtained because this time 64 bits of a complete state candidate are matched with 64 *KES* bits. Afterward, we run the A5/1 in reverse direction to discover the key.

3.3. Attack Analysis. After initialization, we have state candidates with all bits of *REG*1 (guessed register), three bits of *REG*2 (i.e., *REG*2[10], *REG*2[20], and *REG*2[21]), and five bits of *REG*3 (i.e., *REG*3[7], *REG*3[10], *REG*3[20], *REG*3[21], and *REG*3[22]). Out of these eight bits, three bits: MSB of *REG*2, MSB of *REG*3, and second MSB of *REG*3, remain known in further rounds of attack. However, determination algorithm determines full contents of *REG*2 and *REG*3 for those state candidates whose counters c_2 and c_3 have been reached to 10. An important factor in determination algorithm is the saving of 50% state candidates in each round since second MSB of *REG*2 becomes available by the applicability of one of the equations in equations (5)–(7). That is, whenever *REG*2 is clocked, there is a one-half reduction of state candidates in each round. In equations (5)–(7), *i* is initialized to "0" and increases by "1" in every additional clocking.

$$\text{REG2}[20] = \text{REG1}[17] \oplus \text{REG3}[21] \oplus \text{KES}[i+1], \quad (5)$$

$$\text{REG2}[20] = \text{REG1}[17] \oplus \text{REG3}[22] \oplus \text{KES}[i+1], \quad (6)$$

$$\text{REG2}[20] = \text{REG1}[18] \oplus \text{REG3}[21] \oplus \text{KES}[i+1]. \quad (7)$$

An interesting point about *REG*2 is that whenever it is clocked, its clocking bit and 20th bit (simultaneously) become unknown. Therefore, we need to assign both (0 and 1) possibility to *REG*2[10] by replicating the register *REG*2 and this concept is repeated till $1 \le c_2 < 10$, while in the case of *REG*3, during the first two rounds of clocking, we guess index bits 7, 10, and 20. During and in between 3rd to 8th clocking, we guess index bits 7 and 20. After 8th and before 10th clocking, we guess only *REG*3[20]. Here, we assign both (0 and 1) possibilities to index bits by replicating *REG*3. Table 1 shows all possible cases for 10th and 20th index bits of *REG*2 and *REG*3 being known. The attacking phenomenon saves 50% cases for almost all possible cases. Since cases such as *REG*2[10] is known and *REG*2[20] is unknown, and REG2 [10] is unknown and REG2 [20] is known are not applicable during $1 \le c_2 < 10$, such cases need not to be mentioned in Table 1.

3.3.1. Time Complexity. The computational complexity of the proposed attack depends on the number of replications of registers after each A5/1 clocking, where one replication of state candidate takes one unit of time. Without loss of

```
Input: 64-bit known keystream KES and initial state candidates.
Initialize clock counters: $c_2 = 0$ and $c_3 = 0$.
Repeat for each instance of state candidate
step1: computation based on majority rule
if all three registers are clocked then compute REG2[20] using equation (5), increment counters $c_2$++, $c_3$++, and go to step2
else if registers REG1 and REG2 are clocked then compute REG2[20] using equation (6), increment counter $c_2$++, and go to step3
else if registers REG2 and REG3 are clocked then compute REG2[20] using equation (7), increment $c_2$++, $c_3$++, and go to step2
else increase counter $c_3$++, and go to step2
end if
step2:
if $c_3 < 8$ then
assign both possibility (i.e., 0 and 1) to REG3[20] and REG3[7] by replicating REG3
else if $8 \le c_3 < 10$ then assign both possibility to REG3[20] by replicating REG3
end if
step3:
if clocking bit of REG2 is known and unknown for REG3 then make two replicas: copy 1: REG3[10] = 0; copy 2: REG3[10] = 1;
else if clocking bit of REG3 is known and unknown for REG2 then make two replicas:
                copy 1: REG2[10] = 0; copy 2: REG2[10] = 1;
else make four replicas:
                copy 1: REG2[10] = REG3[10] = 0; copy 2: REG2[10] = REG3[10] = 1;
                copy 3: REG2[10] = 0, REG3[10] = 1; copy 4: REG2[10] = 1, REG3[10] = 0;
end if
Until ($c_2 \ge 10$ and $c_3 \ge 10$)
```

Algorithm 1: Determining registers *REG*2 and *REG*3.

Table 1: Percentage saving in finding unknown bit.

*REG*2[10]	*REG*2[20]	*REG*3[10]	*REG*3[20]	Number of cases	Valid cases	Saving cases (%)
Known	Known	Known	Known	0	0	0
Known	Known	Unknown	Unknown	4	2	50
Unknown	Unknown	Known	Known	4	2	50
Unknown	Unknown	Unknown	Unknown	16	8	50
Unknown	Unknown	Known	Unknown	8	4	50
Known	Known	Known	Unknown	2	2	0

generality, we assume that *REG*2 and *REG*3 are clocked 10 times simultaneously.

(a) Due to the condition {1} or {2}, we save 50% state candidates, and due to equation (3) or equation (4), we again save 50% state candidates. In this way, we get $2^8 * (1/2)^2 = 2^6$ initial state candidates. During the run of determination algorithm, in the first two rounds, 2^4 replicated copies are created, 2^3 replicate copies are needed after 3rd and before 8th round, and 2^2 replicated copies are created before 10th clock. Consequently, the time complexity for each guess of *REG*1 is computed as $2^6 * (2^4)^2 * (2^3)^5 * (2^2)^2 = 2^{33}$. Finally, the worst case time complexity $= 2^{19} * 2^{33} = O(2^{52})$, where 2^{19} is associated with guessing of 2^{19} possible state candidates of *REG*1.

(b) Time complexity can be computed in a different way as well. As shown in Table 1, in each round, either *REG*2 is clocked with *REG*3 or not, we are saving half of the possible cases over exhaustive search. Since *REG*2 needs to be clocked 10 times, we save $(1/2)^{10}$ cases. Moreover, initially, due to condition {1} or {2}, we save 50% state candidates, and due to equation (3) or equation (4), we again save 50% state candidates. In this way, we get the reduction in state candidates by $(1/2)^2$. Hence, the worst case time complexity to get the key $= 2^{64} * (1/2)^2 * (1/2)^{10} = O(2^{52})$.

From (a) and (b), we conclude that the computational complexity of the proposed attack in the worst case is $O(2^{52})$.

3.3.2. Success Probability. Since the amount of complete state candidates rises with increasing rounds, the success probability of the attack also rises. Consequently, as all the state candidates are complete, the probability of successful determining the whole key becomes high. An important factor in success analysis is that during each clocking of A5/1, at least two registers are clocked, i.e., a register is clocked 3 out of 4 times.

(1) Probabilistic Analysis. In this analysis, we find the "number of expected rounds" for achieving a higher rate of success (about 90% to 100%).

Formally, we denote

$n_1 \longrightarrow$an event when $REG2$ and $REG3$ are clocked together.

$n_2 \longrightarrow$an event when $REG1$ is clocked either with $REG2$ or $REG3$.

$\text{Prob}(n_1) \longrightarrow$probability of event n_1 occurs which is (1/2).

$\text{Prob}(n_2) \longrightarrow$probability of event n_2 occurs which is (1/2).

Let $\text{Prob}(n_2) = \text{Prob}(n_2') + \text{Prob}(n_2'') = (1/4) + (1/4) = 1/2$, where $\text{Prob}(n_2')$ is the probability of occurring the event: $REG1$ is clocked with $REG2$ $\text{Prob}(n_2'')$ is the probability of occurring the event:$REG1$ is clocked with $REG3$ Thus, $\text{Prob}(n_1) + \text{Prob}(n_2') + \text{Prob}(n_2'') = 1$.

During attack analysis, we observed that to determine all bits of $REG2$ and $REG3$, both registers must be clocked 10 times (at least). This implies that $x_2 = 10$ and $x_3 = 10$. Let us consider $x_1 = 10$. Thus, the expectation $E[Y]$ can be given as follows:

$$E[Y] = x_1 * \text{Prob}(n_1) + x_2 * \text{Prob}(n_2') + x_3 * \text{Prob}(n_2'')$$
$$\text{Prob}(n_1) + \text{Prob}(n_2') + \text{Prob}(n_2''), \tag{8}$$
$$E[Y] = \frac{10 * (1/2) + 2 * (10 * (1/4) + 10 * (1/4))}{(1/2) + (1/4) + (1/4)} = 15,$$

where we abbreviate expected number of rounds by $E[Y]$ and x_1, x_2, and x_3 are counts of clocking cycles for events n_1, n'_2, and n''_2, respectively.

(2) Experimental Analysis. Number of round estimation (as per above calculation) is rough for achieving a higher rate of success; it may be a little bit less or more. Theoretically, it seems impossible to evaluate the exact number of rounds to achieve 100% successful attack. Therefore, we performed some experiments by generating random keys and frames. We mounted the proposed attack for each guess of $REG1$. Also, tapping bits were guessed whenever required during the cryptanalytic attack. The attack was tested on A5/1 for 15 rounds and in parallel, all replicated 2^{33} state candidates were determined and verified. We found that the experimental results corroborate with our probabilistic analysis. Table 2 shows that the success rate of the attack is on average 96.6% in 15 clocking rounds. We performed the experiments on an Intel Quad-Core PC i7@3.40 Ghz 1000 times, and results demonstrated that the session key can be obtained in an average of 12 seconds (7 seconds are required to derive 2^{33} state candidates and 5 seconds are required to verify the search candidates).

3.4. Space Complexity. For each guess, 2^{33} state candidates are needs to be verified. For verification, memory requirement can be determined as $2^{33} * 64$ bits $= 2^{36}$ bytes $= 2^6$ GB, i.e., 64 GB RAM. If the key is not matched for this choice, in the next guess of 19 bits of $REG1$, previous guess is discarded, and the verification of the new state candidate is initiated. In this way, the proposed attack requires a total of only 64 GB RAM.

Table 2: Statistical results for the successful attacks based on the experimental analysis (η_{tsc}: mean of total state candidates; η_{csc}: mean of complete state candidates).

Number of rounds	η_{tsc}	η_{csc}	$(\eta_{tsc}/\eta_{csc}) * 1000$ (%)
10	$2^{28.21}$	$2^{24.12}$	5.87
11	$2^{29.50}$	$2^{26.45}$	12.07
12	$2^{30.30}$	2^{28}	20.30
13	$2^{31.80}$	$2^{30.20}$	32.98
14	$2^{32.12}$	$2^{31.45}$	62.85
15	2^{33}	$2^{32.95}$	96.60

4. Related A5/1 Attacks and Their Comparison

In this section, we discussed only those previous attacks that were proposed on the exact design of A5/1, practically efficient, and which belong to the GD class of attacks.

For attacking A5/1, Biham and Dunkelman's [13] idea is to wait for an event that can leak a huge volume of information about the key and then exploit it. The basic attack assumes that the register $REG3$ has not been clocked for 10 consecutive rounds. The idea is that if we know the clocking bit of $REG3$, we can get the information about clocking bit of $REG1$ and $REG2$ for all 10 rounds because both registers are considered the complement of $REG3[10]$. Formally, KES[i] $\oplus REG3[22]$ is equal to the output bits of $REG1 \oplus REG2$. In this way, guessing of following bits uncover all bits of REG1 and REG2: REG3 [22], REG2 [0], REG1 [9] to REG1 [12], and REG1 [14] to REG1 [18]. In brief, the total complexity is the sum of the cost of guessing 11 bits as mentioned above, the cost of guessing 11 bits of $REG3$ (i.e., $REG3[0]$ to $REG3$ [10]), and the cost of continuous clocking of the third register, so that the attacker receives all unknown bits of $REG3$. Additionally, the attacker needs to probe about 2^{20} different locations by hit and trial to find information revealing events where $REG3$ is not clocked 10 times consecutively. The computational complexity of the attack is 2^{47}, but the attack needs $2^{20.8}$ KES bits. This basic attack was further improved in [13] that reduces the time complexity to $2^{39.91}$, but it also requires a lot of precomputed data, big amount of known KES bits, and large space. Hence, practically such attacks are very difficult to implement.

Biryukov et al. [15] proposed two similar nongeneric TMDTO class attacks on the A5/1 by exploiting low sampling resistance properties of A5/1. Since these attacks belong to TMDTO class, there are many possible choices of trade-off, and three of them are summarized in Table 3, where the success probability of each attack is 60%. Even though the random subgraph attack has less data complexity than biased birthday attack, fetching keystream for 2 seconds, i.e., the requirement of approximately 25000 known KES bits, is a big obstacle to practically implement such an attack. Barkan et al. [17] proposed "ciphertextonly" attack which is an important contribution in the TMDTO class. An important fact about GSM is that before encryption, GSM employs error correction, and such weakness has been exploited by the authors to attack A5/1. However, this attack requires precomputation of the huge volume of data, and those data must be stored in secondary storage; therefore, it

Table 3: Comparison of related attacks (PW: precomputation workload; CC: computational complexity).

Attack	PW	CC	Known keystream	Success probability (%)	Memory requirement	Time to recover key
Proposed in [12]	2^{38}	$2^{39.91}$	$2^{20.8}$ bits	63	64 GB	Impractical attack
Biased birthday attack 1 [14]	2^{48}	1 second	$2^{20.5}$ bits	60	146 GB	Impractical attack
Biased birthday attack 2 [14]	2^{42}	1 second	$2^{20.5}$ bits	60	292 GB	Impractical attack
Subgraph attack [14]	2^{48}	3–6 minutes	$2^{14.7}$ bits	60	146 GB	Impractical attack
Proposed in [15]	0	$2^{51.24}$	Only 64 bits	18	0	7 hours
Proposed in [20]	0	$2^{54.04}$	Only 64 bits	100	0	7 hours
Proposed in [4]	NA	NA	Only 64 bits	81	984 GB	9 seconds
Proposed in [5]	NA	NA	Only 64 bits	80.08	3.84 TB	33 seconds
Proposed in this paper	0	2^{52}	Only 64 bits	96.6	64 GB	12 seconds

Large amount of keystream requirements makes first four attacks impractical; needing of lot of precomputation data is also a big obstacle in practicality of these four attacks.

is very difficult to make such attacks practical. It is noteworthy that simple forms of TMDTO class of attacks have been independently proposed by Babbage [9] and Golić [10]. Keller and Seitz [16] demonstrated a hardware-based attack which differs from previous approaches in two aspects. First, the attack needs a very small amount of plaintext (only 64 bits). Second, the attack is not solely based on the software because its crucial part is implemented on field-programmable gate array (FPGA). A relation between S, S', and S'' is shown in Figure 2.

Keller and Seitz's algorithm is a two-phase backtracking algorithm, where the first phase tries to generate state S' from few states of S and then generates output keystream. Afterward, the second phase computes state S'' corresponding to state S'. The state S'' can be easily generated because the frame number is known, and all registers were clocked linearly. It should be noted that the basic part of Keller and Seitz's algorithm is like Anderson's [1] algorithm. Unlike Anderson's approach, the asynchronous clocking of A5/1 is also considered by Keller and Seitz. The basic strategy is to guess all possible, i.e., 241 states of $REG1$ and $REG2$, and then deduce the largest register $REG3$ as follows. Foremost, i.e., at time $t = 0$, the MSB of $REG3$ is obtained by XORing MSB of $REG1$, MSB of $REG2$, and first known KES bit.

Afterward, Keller and Seitz guessed the clocking bit of $REG3$ by inspecting clocking bit of $REG1$ and $REG2$. If clocking bit of $REG1$ and $REG2$ is not equal, then $REG3$ is clocked either with $REG1$ or $REG2$, i.e., both possibilities of $REG3$ clocking bit are checked. However, if clocking bit of $REG1$ and $REG2$ is the same, then at least these two registers are clocked, i.e., all three registers may have been clocked. Nevertheless, during extensive analysis of this attack, Gendrullis et al. [21] observed that Keller and Seitz's algorithm reduces the complexity of a simple GD attack by identifying contradiction early that can occur by guessing the $REG3$ clocking bit so that $REG3$ will not be clocked. In other words, the authors discard a substantial fraction of potential candidates at an early stage of the verification process. Moreover, Keller and Seitz claim that their attack technique reduces the worst case complexity to $2^{51.24}$ including expected 14 clock cycles per guess. Unfortunately, the approach given by Keller and Seitz [16] not only eliminates wrong candidates but also restricts the search for correct candidates. It should be noted that Keller and Seitz did not mention this fact. Furthermore, no complete analysis of the attack has been performed. The expected computing time for a guess is slightly higher than that stated by Keller and Seitz. The number of all valid state candidates for one fixed guess of $REG1$ and $REG2$ is $(1+(3/4))^{11} = (7/4)^{11} = 2^{8.88} = 471$. Also, the success rate of the attack is only 18% ((86/471) = 0.18) [21].

A significant modification to the above-stated attack is presented by Gendrullis et al. [21]. Unlike Keller and Seitz, they checked all possibilities of clocking bit of $REG3$. It implies that they have considered the cases of clocking and un-clocking of $REG3$ to find the correct state candidates. This decreases 0.25 of all cases and the number of choices from two to one. In this way, the expected number of possibilities for $REG3$ that needs to be verified is roughly 471 for each of the fixed guesses of $REG1$ and $REG2$. The attack has computational complexity of $2^{54.02}$. The attack was tested on the Cost-Optimized Parallel COde Breaker (COPACOBANA) with the knowledge of only 64 KES bits [20]. As a result, the machine can recover the corresponding 64-bit internal state of the cipher in about 6 hours. In this way, the attack procedure significantly reduces the search space over exhaustive search (O (2^{64})). The attack is practically efficient with 100% success. The only limitation is that the COPACOBANA hardware is very expensive and specifically designed for mounting such an attack.

A comparison table for above-discussed attacks is shown in Table 3. It can be clearly observed from the table that there is competition between low keystream attacks which is practically possible, i.e., between those attacks that are proposed by Keller and Seitz [16], Gendrullis et al. [21], Lu et al. [4], Li [5], and the one which is presented in this work. Table 3 shows that the proposed low keystream attack requires 64 GB memory to save replicated state candidates; however, a considerable amount of computational time is reduced. Memory requirement is also very less as compared to the attack proposed by Lu et al. [4] and Li [5]. It should be noted that we are guessing very fewer state candidates, and for each guess, the proposed practical attack takes lesser time, i.e., only 12 seconds with high rate of success (96.6%) which is better as compared to the recently proposed attack by Lu et al. [4] and Li [5].

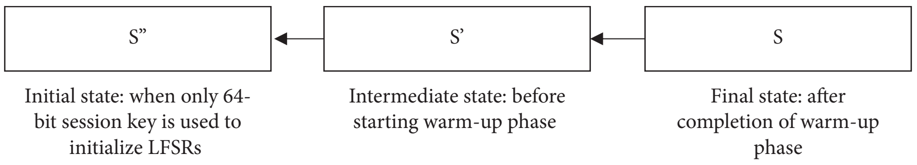

S”
S’
S
Initial state: when only 64-bit session key is used to initialize LFSRs
Intermediate state: before starting warm-up phase
Final state: after completion of warm-up phase

FIGURE 2: Relation between S, S', and S''.

5. Conclusion and Future Work

In this work, we have presented a new low keystream GD attack on the GSM encryption algorithm. The worst case computational complexity of the proposed attack is $O(2^{52})$ which is much better than the brute force ($O(2^{64})$). As evident from the results, the proposed attack is comparable to previous A5/1 attacks in terms of computational complexity and success probability. Since we have considered all the valid state candidates, the probability of containing the secret key by the final set of complete state candidates is high. Through probabilistic analysis and experimental test, we demonstrate that the success rate of the proposed attack is 96.6% and the proposed algorithm is practical and able to recover the key in 12 seconds. The proposed algorithm uses only very small amount of information (64 bits of keystream) to mount the attack. Therefore, the proposed algorithm can be used as a valid and efficient alternative for cryptanalyzing A5/1 and the similar stream ciphers, for example, 64-bit GMR-2 stream cipher which is used in satellite phones. As a future work, we plan to test the proposed algorithm on GMR-2 stream cipher.

Data Availability

No dataset were used to carry out this work.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] R. Anderson, “A5/1: hacking digital phones,” 1994, https://yarchive.net/phone/gsmcipher.html.
[2] Q. Jeremy, “Security in the gsm system,” *AusMobile*, vol. 1, pp. 1–26, 2004.
[3] A. Y. Korkusuz, “Security in the gsm network,” *Term project*, vol. 2012, pp. 1–34, 2012.
[4] J. Lu, Z. Li, and M. Henricksen, “Time–memory trade-off attack on the gsm A5/1 stream cipher using commodity GPGPU,” in *Proceedings of the 14th international conference on applied cryptography and network security*, pp. 350–369, Springer, London, UK, June 2016.
[5] Z. Li, “Optimization of rainbow tables for practically cracking gsm A5/1 based on validated success rate modelling,” in *Proceedings of the 25th international RSA conference*, pp. 359–377, San Francisco, CA, USA, Febraury 2016.
[6] A. Bogdanov, L. R. Knudsen, G. Leander et al., “PRESENT: an ultra-lightweight block cipher,” in *Proceedings of the international workshop on cryptographic hardware and embedded systems*, pp. 450–466, Vienna, Austria, September 2007.
[7] D. Chowdhury, A. Dey, R. Garai et al., “DeCrypt: a 3DES inspired optimised cryptographic algorithm,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, pp. 1–11, 2022.
[8] Gsma, “Zuc specification,” 2012, http://www.gsma.com/technical-projects/wpcontent/uploads/2012/04/eea3eia3zucv16.pdf%202012.
[9] S. Babbage, “A space/time trade-off in exhaustive search attacks on stream ciphers,” *European convention on security and detection*, vol. 408, no. 1, pp. 161–166, 1995.
[10] J. D. Golić, “Cryptanalysis of alleged A5 stream cipher,” in *Proceedings of the advances in cryptology conference EUROCRYPT*, pp. 239–255, Springer, Konstanz, Germany, August 1997.
[11] M. Briceno, I. Goldberg, and D. Wagner, “A pedagogical A5/1 implementation,” *Scientific & Academic*, vol. 1, 1999.
[12] A. Alhamdan, H. Bartlett, E. Dawson, L. Simpson, and K. K. Wong, “Weak key-iv pairs in the a5/1 stream cipher,” in *Proceedings of the 12th Australasian information security conference*, vol. 149, no. 1, pp. 23–36, Sevilla, Spain, December 2014.
[13] E. Biham and O. Dunkelman, “Cryptanalysis of the A5/1 GSM stream cipher,” in *Proceedings of the 1st international conference on cryptology in India INDOCRYPT 2000*, pp. 43–51, Calcutta, India, December 2000.
[14] T. Pornin and J. Stern, “Software-hardware trade-offs: application to A5/1 cryptanalysis,” in *Proceedings of the international workshop on cryptographic hardware and embedded systems CHES*, pp. 318–327, Springer, Massachusetts, MA, USA, April 2000.
[15] A. Biryukov, A. Shamir, and D. Wagner, “Real time cryptanalysis of A5/1 on PC,” in *Proceedings of the international workshop on fast software encryption FSE 2001*, pp. 1–18, Yokohama, Japan, April 2001.
[16] J. Keller and B. Seitz, “A hardware-based attack on the A5/1 stream cipher,” *ITG FACHBERICHT*, vol. 1, pp. 155–158, 2001.
[17] E. Barkan, E. Biham, and N. Keller, “Instant ciphertext-only cryptanalysis of GSM encrypted communication,” in *Proceedings of the 23rd international conference on cryptology CRYPTO 2003*, pp. 600–616, Springer, California, CA, USA, August 2003.
[18] P. Ekdahl and T. Johansson, “Another attack on A5/1,” *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 284–289, 2003.
[19] A. Maximov, T. Johansson, and S. Babbage, “An improved correlation attack on A5/1,” in *Proceedings of the in international workshop on selected areas in cryptography SAC*, pp. 1–18, Kingston, Canada, August 2004.
[20] E. Barkan and E. Biham, “Conditional estimators: an effective attack on A5/1,” in *Proceedings of the in international workshop on selected areas in cryptography SAC 2005*, pp. 1–19, Kingston, Canada, August 2005.

[21] T. Gendrullis, M. Novotný, and A. Rupp, “A real-world attack breaking A5/1 within hours,” in *Proceedings of the international workshop on cryptographic hardware and embedded systems CHES 2008*, pp. 266–282, Washington, DC, USA, August 2008.

[22] M. Kalenderi, D. Pnevmatikatos, I. Papaefstathiou, and C. Manifavas, “Breaking the GSM A5/1 cryptography algorithm with rainbow tables and high end FPGAS,” in *Proceedings of the 22nd international conference on field programmable logic and applications FPL 2012*, pp. 747–753, Belfast, UK, August 2012.

[23] R. Li, H. Li, C. Li, and B. Sun, “A low data complexity attack on the gmr-2 cipher used in the satellite phones,” in *Proceedings of the international workshop on fast software encryption FSE 2014*, pp. 485–501, London, UK, March 2014.

[24] B. Zhang and D. Feng, “New guess-and-determine attack on the self-shrinking generator,” in *Proceedings of the international conference on the theory and application of cryptology and information security ASIACRYPT 2006*, pp. 54–68, Shanghai, China, December 2006.

[25] V. Manjula and C. Chellappan, “A replication attack in wireless sensor networks: analysis and defences,” in *Proceedings of the international conference on computer science and information technology ICCSIT 2011*, pp. 169–178, Chengdu, China, June 2011.

[26] A. D. Dwivedi, P. Borowiecki, and S. Wójtowicz, “Differential-linear and impossible differential cryptanalysis of round-reduced scream,” in *Proceedings of the 14th international joint conference on e-buisness and telecommunications SECRYPT 2017*, pp. 501–506, Madrid, Spain, July 2017.