International Journal of Electrical and Computer Engineering (IJECE)
Vol. 15, No. 6, December 2025, pp. 5453~5465
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i6.pp5453-5465

Memoryless state-recovery cryptanalysis method for lightweight stream cipher – A5/1

Khedkar Aboli Audumbar[1,2], Uday Pandit Khot[3], Balaji G. Hogade[1]
[1]Department of Electronics Engineering, Terna Engineering College, Navi Mumbai, India
[2]Department of Electronics and Telecommunication, Pillai College of Engineering, New Panvel, India
[3]Department of Electronics and Telecommunication Engineering, St. Francis Institute of Technology, Mumbai, India

Article Info

Article history:

Received Jan 29, 2025
Revised Aug 22, 2025
Accepted Sep 16, 2025

Keywords:

Cryptanalysis
Guess-and-determine attack
Time-complexity
GSM
A5/1

ABSTRACT

Cryptology refers to the discipline concerned with securing communication and data in transit by transforming it into an unintelligible form, thereby preventing interpretation by unauthorized entities. Cryptanalysis is the study and practice of analyzing cryptographic systems with the aim of uncovering their weaknesses, finding vulnerabilities and obtaining unauthorized access to encrypted data. A5/1 is a lightweight stream cipher used to protect GSM communications. There are two memoryless cryptanalysis techniques used for this cipher which are Golic's Guess-and-determine attack and Zhang's Near Collision attack. In this paper a new guessing technique called move guessing technique used to construct linear equation filter along with Golic's guess and determine technique is studied. Two modifications in move guessing technique are proposed for recovery of internal states S0 and S1. Further, a novel algorithm is proposed to select the modification to get minimum time complexity for recovery of internal states S0 and S1. The proposed algorithm gives minimum time complexity of $2^{29.3138}$ at $t = 14$ for recovery of S0 state and $2^{43.246}$ for recovery of S1 at $t = 22$.

Corresponding Author:

Khedkar Aboli Audumbar
Department of Electronics Engineering, Terna Engineering College
Plot No 12, Sec-22 Opp. Nerul Railway Station W, Phase II, Nerul, Navi Mumbai, Maharashtra 400706, India
Email: abolikhedkar@gmail.com

1. INTRODUCTION

Now-a-days many applications on internet of things (IoT) and embedded systems are developed. Such application uses global system for mobile communications (GSM) technology for communication. Also, mobile networks use GSM to transmit personal information on radio links [1]. The A5/1 stream cipher are widely used in GSM communication, has been the subject of extensive cryptanalysis since its inception. Ever since its proposal, A5/1 has been attacked with various cryptanalytic methods such as Satisfiability (SAT)-based cryptanalysis, time/memory/data trade-off attacks, guess-and-determine attacks, near collision attack (NCA), ciphertext attack, quantum attack on reduced version of Cipher [2]–[11] [12], [13]. Rainbow Table generation method used to perform time/memory/data trade-off (TMDTO) attack also has high chances of collision [14]. Most of the practical attacks on A5/1 require large, precomputed rainbow table which significantly increases the memory complexities [15]–[18].

Memoryless cryptanalysis techniques on the A5/1 cipher involve analyzing the cipher without relying on previous states or memory. These techniques aim to break the encryption by examining the algorithm's structure and properties without considering historical data. By studying the A5/1 algorithm, cryptanalysts can uncover vulnerabilities and weaknesses that could potentially compromise its security.

Existing Memoryless cryptanalysis techniques namely guess and determine attack and near collision attack that determines S1 recovery state and S0 recovery state, respectively. Guess and determine attack highly relies on identifying patterns and relationships between key and ciphertext [19]. A new low keystream guess-and-determine (GD) attack was proposed in [20] gives the time complexity of 2^{52}. This complexity is higher than the Golic's GD attack's time complexity of $2^{43.15}$ [21]. Zhang's near collision attack recover's S0 state with the time complexity of $2^{53.19}$ [8]. Xu *et al.* further worked on Golic's GD attack and Zhang's near collision attack and was found that the complexity of Golic's S1 recovery attack is no lower than $2^{46.04}$ but higher than the previously believed 2^{43}. On the other hand, Zhang's near collision attack recovers S0 with the time complexity $2^{53.19}$: such a complexity can be further lowered to $2^{50.78}$ with the help of move guessing technique [22]. The 2-bit move guessing based guess and determine attack on A5/1 that can recover internal S0 and S1 state with the complexities of $2^{43.92}$ at t=21 and $2^{43.25}$ at t= 22, respectively [22].

This research focuses on enhancing the cryptanalysis of the A5/1 stream cipher by refining the move guessing technique used to recover internal states. While prior work, such as Golic [21] and the move guessing technique to keep [8] as fixed clock bit method [22], explored partial dependencies among clock bits, the research gap lies in the lack of understanding and systematic treatment of the behavior when other clock bits are held constant in the stop-and-go mechanism. This study systematically investigates the behavior of the cipher when any one out of three clock bits are held constant, revealing how such configurations affect the stop-and-go mechanism. Two novel modifications are proposed in the move guessing technique allowing for effective recovery of internal states S0 and S1. A dynamic decision algorithm is also implemented in these modified techniques to give minimum time complexity.

Although these innovations improve upon existing cryptanalytic strategies for A5/1, they can also offer a framework that can be generalized to other linear feedback shift register (LFSR)-based stream ciphers. The study paves the way for future research into adaptive cryptanalysis and lightweight cipher design, especially in applications where computational efficiency is critical, such as embedded and IoT systems.

This paper is organized as follows. Section 2 provides brief information on key stream generation procedures in A5/1 cipher. Section 3 discusses the Golic's Guess and Determine attack [21] and gives a brief review of 2-bit move guess and determine attack [22]. Later, the modifications in the move guessing technique are introduced in section 4 and proposed an algorithm in section 5 to select the proper modified technique for minimum time complexity. Results and Discussion are done in section 6. Section 7 concludes the research work.

2. THE KEY-STREAM GENERATION PROCEDURE IN A5/1

A5/1 was designed to provide over-the-air communication privacy for the GSM cellular telephone standard and has been widely used in GSM telephony in Europe and the USA. In its design, three combined LFSRs with irregular clocking are used to encrypt bursts of traffic, as is required in GSM [23]. A5/1 has 3-LFSR registers R1, R2 and R3 with sizes 19, 22, 23 respectively making it 64-bit internal state as shown in Figure 1. These states at time t (t = 0, 1, 2….) is represented as [22]:

$$\begin{aligned} S^t &= (R1^t, R2^t, R3^t) \\ &= (S^t[0, \ldots 18], S^t[19, \ldots 40], S^t[41, \ldots 63]) \\ &= (R1^t[0, \ldots 18], R2^t[0, \ldots 21], R3^t[0, \ldots 22]) \end{aligned} \quad (1)$$

Before generating the output bit z^t, A5/1 round function will update the internal state $s^t \rightarrow s^{t+1}$ in a stop-and-go manner as follows:

– Compute maj^t as

$$\begin{aligned} maj^t &= (R_1^t[8] \cdot R_2^t[10]) \oplus (R_1^t[8] \cdot R_3^t[10]) \oplus (R_2^t[10] \cdot R_3^t[10]) \\ &= (s^t[8] \cdot s^t[29]) \oplus (s^t[8] \cdot s^t[51]) \oplus (s^t[29] \cdot s^t[51]) \end{aligned} \quad (2)$$

where '.' is the AND of 2-bits

– If $R_1^t[8] = s^t[8] \neq maj^t$, $R_1^{t+1} \leftarrow R_1^t$, otherwise, call UpdateR1 as follows

$$\begin{aligned} &R_1^{t+1}[i] \leftarrow R_1^t[i-1] \qquad i \in [1,18] \\ &R_1^t[18] \oplus R_1^t[17] \oplus R_1^t[16] \oplus R_1^t[13] \end{aligned} \quad (3)$$

– If $R_2^t[10] = s^t[29] \neq maj^t$, $R_2^{t+1} \leftarrow R_2^t$, otherwise, call UpdateR2 as follows:

$$R_2^{t+1}[i] \leftarrow R_2^t[i-1]\ i \in [1,21]$$
$$\leftarrow R_2^t[21] \oplus R_2^t[20] \quad (4)$$

– If $R_3^t[10] = s^t[51] \neq maj^t$, $R_3^{t+1} \leftarrow R_3^t$, otherwise, call UpdateR1 as follows:

$$R_3^{t+1}[i] \leftarrow R_3^t[i-1]\ \ i \in [1,22]$$
$$\leftarrow R_3^t[22] \oplus R_3^t[21] \oplus R_3^t[20] \oplus R_3^t[7] \quad (5)$$

Then the Output keystream bit z^t is generated as:

$$z^t = R_1^{t+1}[18] \oplus R_2^{t+1}[21] \oplus R_3^{t+1}[22]$$
$$z^t = s^{t+1}[18] \oplus s^{t+1}[40] \oplus s^{t+1}[63] \quad (6)$$

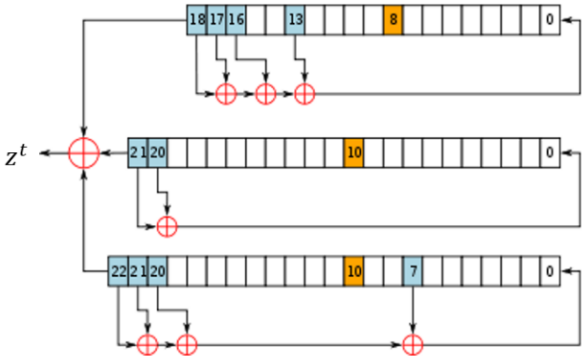

Figure 1. A5/1 stream cipher [24]

3. EXISTING ATTACK FOR A5/1

Although there are many techniques for attacks for A5/1, this paper discusses only the memoryless techniques of attacks such as 1. Guess and Determine attack and 2. Near Collision attack. Near collision attack is challenged in many ways, after the theoretical analysis and practical implementations it is observed that non-randomness claimed in [8] can hardly be achieved so concluded that Near collision attack cannot have less time complexity compared to Guess and Determine attack [25]. And hence, in this paper discussion is done only with Guess-and-Determine attack.

3.1. Golic's guess-and-determine attack [21]

For each step i = 0, 1, 2…, whether the registers $R1$; $R2$; $R3$ are updated or not depends on the three clock bits $s^i[8,29,51]$. Such 3-bit clocks can also be regarded as a 3-bit integer $C^i \in \{0,1 \ldots \ldots 7\}$ defined as in (7).

$$C^i[0,1,2] = s^i[8,29,51] = (p, \rho, \sigma) \quad (7)$$

In Golic's guess-and-determine model, the adversary aims at recovering the initial state s^1, the state right before the generation of z^0. So, the to-be-guessed clocks are C^i for i = 1, 2, …. With the knowledge of c^i, each bit of $\boldsymbol{s}^{i+1}$ can be represented as a linear combination of s^i bits and, following (7), the adversary can deduce three linear equations.

$$s^i[8] = p$$
$$s^i[29] = \rho$$
$$s^i[51] = \sigma$$

From the output z^i, the adversary can further deduce one linear equation.

$$z^i = s^{i+1}[18] \oplus s^{i+1}[40] \oplus s^{i+1}[63]$$

In other words, by guessing 3-bit C^i, the adversary can deduce 4 linear equations of state bits. Golic propose a basic attack that guess 3t clock bits $C^1 \dots \dots C^t$. Based on the t + 1 output bits $z^0 \dots \dots z^t$, the adversary can deduce a system of averaging $1 + 3t + \frac{4}{3}t$ linear equations. For $t \geq 14.38$, the system can involve $1 + 3t + \frac{4}{3}t \geq 63.32$ equations which is sufficient for identifying the correct guess uniquely with "high probability". Although the number of equations and the "high probability" have never been verified, the complexity of Golic's attack is usually believed as $2^{3t} \geq 2^{43.15}$ steps where each step involves the solution of a linear system. Apparently, such a complexity evaluation assumes that the wrong-guess oriented linear equation system acts randomly, and its rank grows linearly with t to 63.32. It is later proved that such an assumption is not true for A5/1.

Besides, Golic also notices that not all 3t clock bits $C^1 \dots \dots C^t$ are to be guessed independently. According to the stop-and-go mechanism, there are occasions where only 2 out of the 3 LFSRs are updated ($C^i \notin \{0, 7\}$) and 1 out of the 3 C^{i+1} bits are already known in C^i. To avoid such redundant bit guesses, Golic propose "branching technique" where a tree structure is applied to track the known bits to further lower the complexity. However, since the branching technique depends on the clock dynamic values, the complexities in did not take the effect of the branching technique into the evaluations. This technique detects S^1 state which may cause delayed process of key detection.

3.2. A brief review of move guessing technique [22]

The 2-bit *move pattern* $m^t \in \{0, \cdots 3\}$ according to the 3-bit clock $C^t = s^t[8,29,51]$. Such a move pattern can be equivalently regarded as 2-dimentional binary vector defined in (8).

$$m^t = m^t[0,1] = (s^t[8] \oplus s^t[29], s^t[8] \oplus s^t[51]) = (\mu, \nu) \in F_2^2 \tag{8}$$

With the knowledge of m^t in above equation, two equations are deduced in (9).

$$\begin{aligned} s^t[8] \oplus s^t[29] &= \mu \\ s^t[8] \oplus s^t[51] &= \nu \end{aligned} \tag{9}$$

The four possible values of m^t, referred as Move 0–3, corresponds to different movements in A5/1 LFSRs transforming s^t to s^{t+1}.

- Move 0: From the LFSR action aspect, updateR1, updateR2 and updateR3 are all called. This corresponds to clock values $C^i \in \{0,7\}$ or equivalently $s^t[8,29,51] \in \{(0,0,0), (1,1,1)\}$
- Move 1: Only updateR2 and updateR3 are called corresponding to $C^i \in \{1,6\}$ or equivalently $s^t[8,29,51] \in \{(0,1,1), (1,0,0)\}$
- Move 2: Only updateR1 and updateR3 are called corresponding to $C^i \in \{2,5\}$ OR $s^t[8,29,51] \in \{(1,0,1), (0,1,0)\}$.
- Move 3: Only updateR1 and updateR2 are called corresponding to $C^i \in \{3,4\}$ OR $s^t[8,29,51] \in \{(1,1,0), (0,0,1)\}$

According to the definition, the LFSR actions before generating the output keystream bits $z^0 \dots \dots z^t$ can be represented as $m^0 \dots \dots m^t$. In this guess and determine attack, first guess the movement m^t corresponding to the transformation $s^t \to s^{t+1}$ and maintains a linear equation set BC by adding new equations according to m^t and the output z^t. For each step t, there are three linear equations: two are from equation (9) according to the move guess and one is from the output z^t as

$$z^t = s^{t+1}[18] \oplus s^{t+1}[40] \oplus s^{t+1}[63] \tag{10}$$

So, each 2-bit move guess results in three equations.

3.2.1. Move guessing based recovering S^0 state

The move equations in (9) and the output equation in (10) correspond to the internal states at different time instances. But this attack is targeted for recovering the initial state s^0. Therefore, the internal states s^t at different time instance t should be represented by s^0 bits for deducing s^0 related equations. The state s^t is deduced from s^0 by taking the moves $m^0 \dots \dots m^t$ as

$$s^0 \xrightarrow{m^0} s^1 \xrightarrow{m^1} \dots \xrightarrow{m^{t-2}} s^{t-1} \xrightarrow{m^{t-1}} s^t \tag{11}$$

The moves $m^0 \dots \dots m^{t-1}$ corresponds to the linear transformations in LFSRs so each s^t bit is a linear combination of s^0 bits: such a linear combination can be regarded as a inner product of s^0 and a 64-bit word $w \in F_2^{64}$. In order to track all state bits in $s^0 \dots . . s^t$ bits, it is defined by 64x64 binary matrices $w^0 \dots \dots w^t \in (F_2^{64})^{64}$ s.t. $s^i = w^i s^0$ for all i = 0, …, t. The row vector of w^i is denoted as $w^i[j]$ for j = 0, …, 63. In this way, the state bit S^i[j] can be computed as the inner produce of initial $\boldsymbol{s}0$ and row vector $w^i[j]$. Each state bit of s^t can be uniformly expressed as a linear combination of s^0 bits as

$$s^t[i] = w^t[i] \cdot s^0, \quad i = 0 \dots 63, \quad t = 0,1,2 \tag{12}$$

For t consecutive movements $m^0 \dots \dots m^{t-1}$ and the corresponding output $z^0 \cdots \cdots z^{t-1}$, the corresponding linear equations set *BC* can be deduced as

$$getBC\big((m^0_{\dots} \cdot m^t - 1), (z^0 \dots z^{t-1})\big) \rightarrow BC$$

Such BC can be regarded as a linear equation system in (13).

$$Ax^T = b^T, \text{ where} \in F_2^{3t \times 64}, x \in F_2^{64}, b \in F_2^{3t} \tag{13}$$

and the solution of the equation above corresponds to all candidate s^0. The number of solutions depends on the rank of the matrix A and its extended matrix as in (14).

$$E = [A, b^T] \tag{14}$$

– If $rank(A) = rank(E)$, there will be $2^{64-\beta_t}$ solutions where β_t is the positive integer defined in (15) as the rank of the matrix A;

$$\beta_t = rank(A) \tag{15}$$

– If $rank(A) \neq rank(E)$, there will no solution at all.

With the guessed moves $m^0 \dots \dots m^{t-1}$ and the observed output bits $z^0 \cdots \cdots z^{t-1}$, we are now able to acquire both A and $\boldsymbol{b}$ along with the extended matrix E in (14).

The probability of $rank(A) = rank(E)$:

– For the correct guess of $m^0 \dots \dots m^{t-1}$, $rank(A) = rank(E)$ is constantly true.
– If $m0$, …, $mt-1$, the probability of $rank(A) = rank(E)$ is defined as $\alpha_t (0 \leq \alpha_t \leq 1)$. According to our analysis, such α_t's grows gradually with t and should be measured practically.

So, the probability of $rank(A) = rank(E)$ can be formally represented as (16).

$$\begin{aligned} P_r[rank(A) = rank(E)] &= 1 \quad && m^0 \dots \dots m^{t-1} \text{ is correctly guessed} \\ &= \alpha_t \in [0,1] && m^0 \dots \dots m^{t-1} \text{ is wrongly guessed} \end{aligned} \tag{16}$$

The general process of such an attack can be summarized as follows:

S1: Guess $m^0 \dots \dots m^{t-1}$, observe $z^0 \cdots \cdots z^{t-1}$ and deduce the linear system represented as $Ax^T = b^T$

S2: Do the rank test check, whether $rank(A) \neq rank(E)$

S3: Traversing the remaining $\boldsymbol{s}0$ candidates and identify the correct s^0 with additional output bits $z^t \cdots \cdots z^{l-1}$ generated by the encryption oracle.

Complexity analysis: in step 3, there are 2^{2t} candidate $(m^0 \dots \dots m^{t-1})$'s. According to (16), averaging $(\alpha_t \cdot 2^{2t})$ move pattern candidates can pass the test. Adding βt in (15), the averaging time complexity can be computed as in (17).

$$\text{Comp} = 2^{2t} + \alpha_t \cdot 2^{2t+64-\beta_t} = 2^{2t} + 2^{2t+64-\beta_t+log\,\alpha_t} \tag{17}$$

By randomly selecting 2^{30} $((m^0 \dots \dots m^{t-1})$, $(z^0 \cdots \cdots z^{t-1}))$ pairs and performing the attack. The value obtained for α_t and β_t for different values of t's are shown in Table 1. The lowest time complexity is $2^{36.56}$ corresponding to t = 16.

For testing purposes, the algorithm given in [21] is executed and the values of the time complexity obtained are shown in Table 1. Although the values obtained are slightly different given in [21] may be because of random key generation, but the pattern obtained is same.

Table 1. The values of α_t and β_t in equation 17 with 2^{30} random tests for S^0 recovery using move guess-and-determine attack

t	β_t	log α_t	log Comp.	t	β_t	log α_t	log Comp.
6	31.954	-0.10	43.93	17	59.4827	-1.96	36.78
7	34.6816	-0.11	43.20	18	60.4008	-2.95	37.36
8	37.5845	-0.11	42.29	19	61.1459	-4.14	38.49
9	40.5501	-0.11	41.33	20	61.7875	-5.49	40.14
10	43.5157	-0.12	40.35	21	62.3641	-7.01	42.03
11	46.4601	-0.15	39.38	22	62.8739	-8.69	44.00
12	49.3675	-0.18	38.44	23	63.2952	-10.52	46.00
13	52.1619	-0.24	37.59	24	63.6082	-12.49	48.00
14	54.6766	-0.36	36.96	25	63.811	-14.55	50.00
15	56.7376	-0.65	36.62	26	63.9225	-16.63	52.00
16	**58.304**	**-1.19**	**36.56**	27	63.9734	-18.64	54.00

3.2.2. Move guessing based recovering S^1 state

For recovering $s1$ according to $z^0 \cdots \cdots z^{t-1}$, we do not need to guess m^0. We guess directly the $t-1$ move patterns $m^1 \ldots \ldots m^{t-1}$ and acquire the linear equation system $Ax^T = b^T$ of sizes $\in F_2^{(2t-1)\times 64}$, $x \in F_2^{64}$, $b \in F_2^{2t-1}$. Therefore, the general process has now become:

S1: Guess moves $m^1 \ldots \ldots m^{t-1}$ and maintain a linear system $Ax^T = b^T$

S2: Do the matrix rank test and discard the wrong guesses satisfying $rank(A) \neq rank(E)$

S3: Traverse the remaining s^1 candidates and identify the correct s^1 with additional output bits $z^t \cdots \cdots z^{l-1}$.

In S1, start from $w^1 = I$ and acquire the bit conditions on $(m^1 \ldots \ldots m^{t-1})$ and $(z^1 \cdots \cdots z^{t-1})$. Besides, letting $s^1 = x = (x_0 \ldots \ldots x_{63})$, there is also an equation deduced from z^0 according to (10) as

$$x_{18} \oplus x_{40} \oplus x_{63} = z^0 \tag{18}$$

Complexity analysis: Among the $2^{2(t-1)}$ moves $m^1 \ldots \ldots m^{t-1}$, there is a portion of α_t passing the rank test and the averaging $rank(A)$ is βt as given in (15). So, the complexity can be evaluated as:

$$\text{Comp} = 2^{2(t-1)} + \alpha_t \cdot 2^{2(t-1)+64-\beta_t} = 2^{2(t-1)} + 2^{2(t-1)+64-\beta_t+log\,\alpha_t} \tag{19}$$

The α_t and β_t parameters are practically evaluated, and the value of complexity are shown in Table 2. The lowest complexity achieved is $2^{43.251}$ at $t = 22$.

Table 2. The values of α_t and β_t in equation 19 with 2^{30} random tests for S^1 recovery using move guess-and-determine attack

t	β_t	log α_t	log Comp.	t	β_t	log α_t	log Comp.
7	19	0	57	18	51.3778	-0.46902	46.153
8	22	0	56	19	53.9439	-0.81697	45.241
9	25	0	55	20	56.3782	-1.28675	44.352
10	28	0	54	21	58.6462	-1.93777	43.545
11	31	0	53	**22**	**60.6179**	**-2.91643**	**43.251**
12	34	0	52	23	62.0883	-4.51626	44.219
13	36.9993	-0.00025	51.000	24	63.0036	-6.73256	46.026
14	39.9919	-0.00514	50.002	25	63.5126	-9.31350	48.003
15	42.9586	-0.03206	49.009	26	63.781	-12.06483	50.000
16	45.8676	-0.09924	48.033	27	63.9129	-14.84823	52.000
17	48.6826	-0.22966	47.087	28	63.9701	-17.57609	54.000

4. PROPOSED MODIFIED MOVE GUESSING TECHNIQUE

Golic has notice that not all 3t clock bits $C^1 \ldots \ldots C^t$ are to be guessed independently. According to the stop-and-go mechanism, there are occasions where only 2 out of the 3 LFSRs are updated ($C^i \notin \{0, 7\}$) and 1 out of the 3 C^{i+1} bits are already known in C^i [12]. But which value of C^{i+1} should be considered that remained constant. 2-bit move pattern in [22] assumed only $s^t[8]$ and calculated the time complexity.

In this paper equation (8) of existing technique has been modified by taking either $s^t[29]$ or $s^t[51]$ constant. Because of these proposed assumptions, move pattern are changed which causes the change in the values of update register and had a wide effect on calculation of time complexity.

4.1. Modification 1- with $s^t[29]$ constant

Equivalently binary vector of dimension 2 defined for $s^t[29]$ is as follows:

$$m^t = m^t[0,1] = (s^t[8] \oplus s^t[29], s^t[29] \oplus s^t[51]) = (\mu 1, \nu 1) \epsilon F_2^2 \quad (20)$$

With the knowledge of m^t in above equation, 2 equations are deduced as follows:

$$s^t[8] \oplus s^t[29] = \mu 1$$
$$s^t[29] \oplus s^t[51] = \nu 1 \quad (21)$$

The 4 possible values of m^t, referred as Move 0–3, corresponds to different movements in A5/1 LFSRs transforming s^t to s^{t+1}.

- Move 0: From the LFSR action aspect, updateR1, updateR2 and updateR3 are all called. This corresponds to clock values $C^i \in \{0,7\}$ or equivalently $s^t[8,29,51] \in \{(0,0,0), (1,1,1)\}$
- Move 1: Only updateR1 and updateR3 are called corresponding to $C^i \in \{1,6\}$ or equivalently $s^t[8,29,51] \in \{(0,1,0), (1,0,1)\}$
- Move 2: Only updateR2 and updateR3 are called corresponding to $C^i \in \{2,5\}$ OR $s^t[8,29,51] \in \{(1,0,0), (0,1,1)\}$.
- Move 3: Only updateR1 and updateR2 are called corresponding to $C^i \in \{3,4\}$ OR $s^t[8,29,51] \in \{(1,1,0), (0,0,1)\}$

As considered in section 3, 3rd equation is considered as

$$z^t = s^{t+1}[18] \oplus s^{t+1}[40] \oplus s^{t+1}[63] \quad (22)$$

4.1.1. Recovery of S^0 state with $s^t[29]$ constant

By modification as discussed in section 4.1 a new code is implemented, and the time complexity is calculated by (17). The result obtained is shown in Table 3. The lowest time complexity achieved is $2^{31.66}$ corresponding to $t = 15$.

4.1.2. Recovery of S^1 state with $s^t[29]$ constant

For recovery of S1 state according to modification made in 4.1 the lowest time complexity is calculated based on (19). The result obtained is shown in Table 4. The lowest time complexity achieved is $2^{43.246}$ corresponding to $t = 22$.

Table 3. The values of α_t and β_t in equation 17 with 2^{30} random tests for S^0 recovery using move guess-and-determine attack

t	β_t	log α_t	log Comp.	t	β_t	log α_t	log Comp.
6	32.286	-3.98	39.72	17	59.5851	-8.89	34.06
7	34.9589	-4.19	38.84	18	60.4835	-10.97	36.00
8	37.8285	-4.28	37.88	19	61.2058	-13.25	38.00
9	40.7774	-4.31	36.90	20	61.8244	-15.62	40.00
10	43.7333	-4.36	35.90	21	62.3833	-18.02	42.00
11	46.6703	-4.42	34.90	22	62.8825	-20.65	44.00
12	49.5674	-4.53	33.90	23	63.2985	-23.32	46.00
13	52.3462	-4.78	32.88	24	63.6093	-26.67	48.00
14	54.8391	-5.24	32.00	25	63.8113	-30	50
15	**56.8772**	**-6.00**	**31.66**	26	63.9225	-29	52
16	58.4244	-7.20	32.40	27	63.9734	-30	54

Table 4. The values of α_t and β_t in equation 19 with 2^{30} random tests for S^1 recovery using move guess-and-determine attack

t	β_t	log α_t	log Comp.	t	β_t	log α_t	log Comp.
7	19	0	57	18	51.3778	-0.46813	46.154
8	22	0	56	19	53.944	-0.81775	45.240
9	25	0	55	20	56.3782	-1.29023	44.349
10	28	0	54	21	58.6461	-1.92793	43.554
11	31	0	53	**22**	**60.618**	**-2.92557**	**43.246**
12	34	0	52	23	62.0883	-4.53347	44.217
13	36.9993	-0.00032	51.000	24	63.0036	-6.74720	46.026
14	39.9919	-0.00664	50.001	25	63.5126	-9.33791	48.003
15	42.9586	-0.02965	49.011	26	63.781	-12.14077	50.000
16	45.8676	-0.09063	48.041	27	63.9129	-15.07695	52.000
17	48.6827	-0.23270	47.084	28	63.9702	-18.01131	54.000

4.2. Modification 2 - with $s^t[51]$ constant

Equivalently binary vector of dimension 2 defined for $s^t[51]$ is as follows:

$$m^t = m^t[0,1] = (s^t[8] \oplus s^t[51], s^t[29] \oplus s^t[51]) = (\mu 2, \nu 2) \epsilon F_2^2 \quad (23)$$

With the knowledge of m^t in above equation, 2 equations are deduced as follows:

$$s^t[8] \oplus s^t[51] = \mu 2$$
$$s^t[29] \oplus s^t[51] = \nu 2 \quad (24)$$

The 4 possible values of m^t, referred as Move 0–3, corresponds to different movements in A5/1 LFSRs transforming s^t to s^{t+1}.

- Move 0: From the LFSR action aspect, updateR1, updateR2 and updateR3 are all called. This corresponds to clock values $C^i \in \{0,7\}$ or equivalently $s^t[8,29,51] \in \{(0,0,0),(1,1,1)\}$
- Move 1: Only updateR1 and updateR2 are called corresponding to $C^i \in \{1,6\}$ or equivalently $s^t[8,29,51] \in \{(0,0,1),(1,1,0)\}$
- Move 2: Only updateR2 and updateR3 are called corresponding to $C^i \in \{2,5\}$ OR $s^t[8,29,51] \in \{(1,0,0),(0,1,1)\}$.
- Move 3: Only updateR1 and updateR3 are called corresponding to $C^i \in \{3,4\}$ OR $s^t[8,29,51] \in \{(1,0,1),(0,1,0)\}$

As considered in section 3, 3rd equation is considered as

$$z^t = s^{t+1}[18] \oplus s^{t+1}[40] \oplus s^{t+1}[63] \quad (25)$$

4.2.1. Recovery of S^0 state with $s^t[51]$ constant

By modification as discussed in section 4.2 a new code is implemented, and the time complexity is calculated by (17). The result obtained is shown in Table 5. The lowest time complexity achieved is $2^{29.313}$ corresponding to $t = 14$.

Table 5. The values of α_t and β_t in equation 17 with 2^{30} random tests for S^0 recovery using move guess-and-determine attack

t	β_t	log α_t	log Comp.	t	β_t	log α_t	log Comp.
6	32.4519	-6.977	36.570	17	59.6558	-13.427	34.002
7	35.0975	-7.128	35.774	18	60.5407	-15.803	36.000
8	37.9506	-7.187	34.862	19	61.2472	-18.352	38.000
9	40.8911	-7.208	33.900	20	61.8501	-20.820	40.000
10	43.8423	-7.243	32.913	21	62.3964	-23.573	42.000
11	46.7768	-7.287	31.937	22	62.888	-25.912	44.000
12	49.6701	-7.405	30.935	23	63.3004	-29	46
13	52.4454	-7.775	29.880	24	63.6098	-30	48
14	**54.9343**	**-8.494**	**29.313**	25	63.8114	-30	50
15	56.9666	-9.626	30.221	26	63.9225	-30	52
16	58.5056	-11.298	32.0255	27	63.9734	-30	54

4.2.2. Recovery of S^1 state with $s^t[51]$ constant

For recovery of S^1 state according to modification made in 4.2 the lowest time complexity is calculated based on equation 19. The result obtained is shown in Table 6. The lowest time complexity achieved is $2^{43.251}$ corresponding to $t = 22$.

Remark: Move guess and determine technique discussed in section 3 and section 4 is based on the Golic's observations stating only 2 out of 3 LFSRs are updated and 1 out of 3 LFSRs always retains its previous state. So in [12], $S[8]$ clock bit is considered common, and the Lowest time complexity result that we obtain to recover S0 and S1 state is $2^{36.56}$ corresponding to $t = 16$ and $2^{43.251}$ at $t = 22$.

We have also obtained the time complexity result by keeping the $S[29]$ and $S[51]$ clock bit state as previous state and observed that time complexity can again be lowered further. To recover S^0 and S^1 state the obtained lowest time complexity is $2^{31.66}$ corresponding to $t = 15$, $2^{43.246}$ at $t = 22$ for $S[29]$ and $2^{29.313}$ corresponding to $t = 14$, $2^{43.251}$ corresponding to $t = 22$ for $S[51]$.

Table 6. The values of α_t and β_t in equation 19 with 2^{30} random tests for S^1 recovery using move guess-and-determine attack

t	β_t	log α_t	log Comp.	t	β_t	log α_t	log Comp.
7	19	0	57	18	51.3778	-0.46835	46.15415
8	22	0	56	19	53.944	-0.81716	45.24122
9	25	0	55	20	56.3782	-1.28782	44.35174
10	28	0	54	21	58.6462	-1.92914	43.55312
11	31	0	53	**22**	**60.618**	**-2.91628**	**43.25156**
12	34	0	52	23	62.0883	-4.53073	44.21757
13	36.9993	-0.00048	51.00021	24	63.0037	-6.74762	46.02653
14	39.9919	-0.00581	50.00228	25	63.5126	-9.35980	48.00307
15	42.9586	-0.02951	49.01188	26	63.781	-12.16150	50.00036
16	45.8676	-0.09465	48.03775	27	63.9129	-15.06589	52.00004
17	48.6826	-0.23243	47.08500	28	63.9701	-18.00176	54.00000

5. ALGORITHM TO SELECT A CONSTANT CLOCK BIT USED FOR GUESS AND DETERMINE TECHNIQUE

According to stop-and-go mechanism in section 2 the occasion when 1 out of 3 clock bits remains as previous, depends on maj^t (2). Using this (2) it has been derived that which of the clock bit will remain same as the previous clock bit and this clock bit is considered constant. Using this constant clock bit move equation m^t of (9), or (21), or (24) will be calculated. Using this m^t, further recovery process of S^0 and S^1 will be done as discussed in section 3 and 4.

Attack procedure: The steps involved in attack procedure are as follows.

The general process of such an attack can be summarised as follows:
Step 1: Compute maj^t from equation (2)
Step 2: Check which clock bit $\neq maj^t$
Step 3: That clock bit is considered common bit in 2-bit Move equation.

If (s[8] != maj^t) **then**
- (a) Acquire ℓ keystream bits $z^0, \ldots, z^{\ell-1}$
- (b) Initialise S $\leftarrow\phi$ for collecting $\boldsymbol{s}^0$ candidates
- (c) Guess $(m^0, \ldots, m^{t-1})$ and do the following sub steps:
  - a. Acquire the equations BC $\leftarrow$ getBC($(m^0, \ldots, m^{t-1})$, $(z^0, \ldots, z^{t-1})$) by calling Algorithm 1.
  - b. Deduce the A and b in (13) according to BC and compute the extended matrix E in (14).
  - c. Compute $rank(A)$ and $rank(E)$, if $rank(A) \neq rank(E)$, such a movement guess is wrong, go back to Step 3 for the next movement guess.
  - d. For all $2^{64-rank(A)}$ solutions to $A\boldsymbol{x}^T=b^T$, set $\hat{s}^0 \leftarrow \boldsymbol{x}$ and generate the keystream bits $\hat{z}^0, \ldots, \hat{z}^{t-1}, \hat{z}^t, \ldots; \hat{z}^{\ell-1}$.
  - e. If $(\hat{z}^t, \ldots, \hat{z}^{\ell-1}) = (z^t, \ldots, z^{\ell-1})$, add such $\hat{s}^0$ into S.
- (d) Return S.

Else If (s[29] != maj^t) **then**
- (a) Acquire ℓ keystream bits $z^0, \ldots, z^{\ell-1}$
- (b) Initialise S $\leftarrow\phi$ for collecting $\boldsymbol{s}^0$ candidates
- (c) Guess $(m^0, \ldots, m^{t-1})$ and do the following sub steps:
  - a. Acquire the equations BC $\leftarrow$ getBC($(m^0, \ldots, m^{t-1})$, $(z^0, \ldots, z^{t-1})$) by calling Algorithm 2.
  - b. Deduce the A and b in (13) according to BC and compute the extended matrix E in (14).
  - c. Compute $rank(A)$ and $rank(E)$, if $rank(A) \neq rank(E)$, such a movement guess is wrong, go back to Step 3 for the next movement guess.
  - d. For all $2^{64-rank(A)}$ solutions to $A\boldsymbol{x}^T=b^T$, set $\hat{s}^0 \leftarrow \boldsymbol{x}$ and generate the keystream bits $\hat{z}^0, \ldots, \hat{z}^{t-1}, \hat{z}^t, \ldots; \hat{z}^{\ell-1}$.
  - e. If $(\hat{z}^t, \ldots, \hat{z}^{\ell-1}) = (z^t, \ldots, z^{\ell-1})$, add such $\hat{s}^0$ into S.
- (d) Return S.

Else If (s[51] != maj^t) **then**
- (a) Acquire ℓ keystream bits $z^0, \ldots, z^{\ell-1}$
- (b) Initialise S $\leftarrow\phi$ for collecting $\boldsymbol{s}^0$ candidates
- (c) Guess $(m^0, \ldots, m^{t-1})$ and do the following sub steps:
  - a. Acquire the equations BC $\leftarrow$ getBC($(m^0, \ldots, m^{t-1})$, $(z^0, \ldots, z^{t-1})$) by calling Algorithm 3.
  - b. Deduce the A and b in (13) according to BC and compute the extended matrix E in (14).
  - c. Compute $rank(A)$ and $rank(E)$, if $rank(A) \neq rank(E)$, such a movement guess is wrong, go back to Step 3 for the next movement guess.
  - d. For all $2^{64-rank(A)}$ solutions to $A\boldsymbol{x}^T=b^T$, set $\hat{s}^0 \leftarrow \boldsymbol{x}$ and generate the keystream bits $\hat{z}^0, \ldots, \hat{z}^{t-1}, \hat{z}^t, \ldots; \hat{z}^{\ell-1}$.
  - e. If $(\hat{z}^t, \ldots, \hat{z}^{\ell-1}) = (z^t, \ldots, z^{\ell-1})$, add such $\hat{s}^0$ into S.
- (d) Return S.

Algorithm 1. Deduce the set of equations according to the given moves and output bits

1. **procedure** getBC (movements (m^0, …, m^{t-1}) $\in \{0,3\}^t$, output bits (z^0,…..,z^{t-1}) $\in F_2^t$
2. Initialise the words $W^0 \leftarrow I$
3. Initialise the linear equations set BC $\leftarrow \phi$
4. Initialise $\boldsymbol{x} =$ (x^0 , …., x^{63}) as vector of 63 unknown Boolean variables corresponding to the 64 state bits of $\boldsymbol{s}^0$
5. **for** i = 0, 1, …, t − 1 **do**
   a. Represent $m^i = (\mu, \nu) \in \{0, …, 3\}$ as Equation (8)
   b. Update BC by adding the following equations
      i. $(w^i[8] \oplus w^i[29]) \cdot x = \mu$
      ii. $(w^i[8] \oplus w^i[51]) \cdot x = \nu$
   c. Deduce w^{i+1} according to w^i by calling $w^{i+1} \leftarrow$UpdW(m^i, w^i) defined in Algorithm 3 ref. [12].
   d. Update BC by adding the following linear equations corresponding to Equation (10)
      i. $(w^{i+1}[18] \oplus w^{i+1}[40] \oplus w^{i+1}[63])\ .\ x = z^i$
6. **End for**
7. **Return** BC
8. **End Procedure**

Algorithm 2. Deduce the set of equations according to the given moves and output bits

1. **procedure** getBC (movements (m^0, …, m^{t-1}) $\in \{0,3\}^t$, output bits (z^0,…..,z^{t-1}) $\in F_2^t$
2. Initialise the words $W^0 \leftarrow I$
3. Initialise the linear equations set BC $\leftarrow \phi$
4. Initialise $\boldsymbol{x} =$ (x^0 , …., x^{63}) as vector of 63 unknown Boolean variables corresponding to the 64 state bits of $\boldsymbol{s}^0$
5. **for** i = 0, 1, …, t − 1 **do**
   a. Represent $mi = (\mu 1, \nu 1) \in \{0, …, 3\}$ as Equation (20)
   b. Update BC by adding the following equations
      i. $(w^i[8] \oplus w^i[29]) \cdot x = \mu 1$
      ii. $(w^i[29] \oplus w^i[51]) \cdot x = \nu 1$
   c. Deduce w^{i+1} according to w^i by calling $w^{i+1} \leftarrow$UpdW(m^i, w^i) defined in Algorithm 3 ref. [12].
   d. Update BC by adding the following linear equations corresponding to Equation (10)
      i. $(w^{i+1}[18] \oplus w^{i+1}[40] \oplus w^{i+1}[63])\ .\ x = z^i$
6. **End for**
7. **Return** BC
8. **End Procedure**

Algorithm 3. Deduce the set of equations according to the given moves and output bits

1. **procedure** getBC (movements (m^0, …, m^{t-1}) $\in \{0,3\}^t$, output bits (z^0,…..,z^{t-1}) $\in F_2^t$
2. Initialise the words $W^0 \leftarrow I$
3. Initialise the linear equations set BC $\leftarrow \phi$
4. Initialise $\boldsymbol{x} =$ (x^0 , …., x^{63}) as vector of 63 unknown Boolean variables corresponding to the 64 state bits of $\boldsymbol{s}^0$
5. **for** i = 0, 1, …, t − 1 **do**
   a. Represent $mi = (\mu 2, \nu 2) \in \{0, …, 3\}$ as Equation (23)
   b. Update BC by adding the following equations
      i. $(w^i[8] \oplus w^i[51]) \cdot x = \mu 2$
      ii. $(w^i[29] \oplus w^i[51]) \cdot x = \nu 2$
   c. Deduce w^{i+1} according to w^i by calling $w^{i+1} \leftarrow$UpdW(m^i, w^i) defined in Algorithm 3 ref. [12].
   d. Update BC by adding the following linear equations corresponding to Equation (10)
      i. $(w^{i+1}[18] \oplus w^{i+1}[40] \oplus w^{i+1}[63])\ .\ x = z^i$
6. **End for**
7. **Return** BC
8. **End Procedure**

Follow the same attack procedure as given above to Recover S^1 state, but no need to guess m^0 move.

6. RESULT AND DISCUSSION

The selective modified Guess and Determine attack algorithm gives the method to select which clock bit is to be considered constant and at every time of executing the attack so that minimum time complexity is achieved to recover S^0 and S^1 state. The practical results obtained for recovery of S^0 and S^1 state of proposed methodology are discussed here.

6.1. Recovery of S^0 state

Modification as discussed is implemented and the time complexity is calculated by (17). The result obtained is shown in Table 7. The lowest time complexity achieved is $2^{29.3138}$ corresponding to $t = 14$.

6.2. Recovery of S^1 state

For recovery of S^1 state according to modification made and the lowest time complexity is calculated based on (19). The result obtained is shown in Table 8. The lowest time complexity achieved is $2^{43.246}$ corresponding to $t = 22$. Time complexity achieved for recovery of S^0 and S^1 state using move guess and determine attack discussed section 3, proposed modified Guess and determine attack considering $S[29]$ and $S[51]$ clock bits from section 4 and selective modified Guess and determine attack section (5) are summerised in Table 9. Table 9 shown that selective modified Guess and determine attack will choose the clock bit which will always give less time complexity.

Table 7. The values of α_t and β_t in equation 17 with 2^{30} random tests for S^0 recovery using move guess-and-determine attack

t	β_t	log α_t	log Comp.	t	β_t	log α_t	log Comp.
6	32.4518	-6.977	36.5708	17	59.6557	-13.433	34.0026
7	35.0974	-7.127	35.7747	18	60.5406	-15.803	36.0002
8	37.9506	-7.187	34.8622	19	61.2472	-18.320	38.0000
9	40.8911	-7.208	33.9006	20	61.8502	-20.991	40.0000
10	43.8423	-7.243	32.9146	21	62.3964	-23.356	42.0000
11	46.7769	-7.287	31.9370	22	62.888	-26.678	44.0000
12	49.6701	-7.407	30.9346	23	63.3003	-28	46.0000
13	52.4454	-7.775	29.8808	24	63.6098	-30	48
14	**54.9342**	**-8.494**	**29.3138**	25	63.8114	-30	50
15	56.9666	-9.626	30.2212	26	63.9225	-30	52
16	58.5056	-11.300	32.0255	27	63.9734	-30	54

Table 8. The values of α_t and β_t in equation 19 with 2^{30} random tests for S^1 recovery using move guess-and-determine attack

t	β_t	log α_t	log Comp.	t	β_t	log α_t	log Comp.
7	19	0	57	18	51.3778	-0.46813	46.154
8	22	0	56	19	53.944	-0.81775	45.240
9	25	0	55	20	56.3782	-1.29023	44.349
10	28	0	54	21	58.6461	-1.92793	43.554
11	31	0	53	**22**	**60.618**	**-2.92557**	**43.246**
12	34	0	52	23	62.0883	-4.53347	44.217
13	36.9993	-0.00032	51.000	24	63.0036	-6.74720	46.026
14	39.9919	-0.00664	50.001	25	63.5126	-9.33791	48.003
15	42.9586	-0.02965	49.011	26	63.781	-12.14077	50.000
16	45.8676	-0.09063	48.041	27	63.9129	-15.07695	52.000
17	48.6827	-0.23270	47.084	28	63.9702	-18.01131	54.000

Table 9. Values of time complexity with various methods for recovery of S^0 and S^1 states

Method	Constant clock bit	Log complexity during recovery of S^0 state	Log complexity during recovery of S^1 state
Move guess and determine attack	S[8]	36.56	43.251
Proposed modified Guess and determine attack	S[29]	31.66	43.246
Proposed modified Guess and determine attack	S[51]	29.313	43.251
Selective modified Guess and determine attack	S[8] / S[29] / S[51]	29.3138	43.246

7. CONCLUSION

Here we propose two modifications in 2-bit move guessing techniques and revisited memoryless state-recovery method move guessing technique and Golic's guess and determine attack on A5/1 stream cipher. With practical implementation we can prove that the time complexity can be further reduced by changing the move equation for recovery of S^0 and S^1. For recovery of S^0 and S^1 time complexity achieved by keeping $S[29]$ bit common is calculated as $2^{31.66}$ corresponding to $t = 15$, $2^{43.246}$ corresponding to $t = 22$ respectively and by keeping $S[51]$ bit common it gives $2^{29.313}$ corresponding to $t = 14$ and $2^{43.251}$ corresponding to $t = 22$. We have also given an algorithm to decide which bit can be kept constant so that for every iteration of finding S^0 and S^1 state bits the time complexity is always at its lower end. Time complexity calculated with this method for recovery of S^0 is $2^{29.3138}$ corresponding to $t = 14$ and for recovery of S^1 is $2^{43.246}$ corresponding to $t = 22$.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support provided by the high-performance systems at the Artificial Intelligence and Machine Learning Laboratory, established through the AICTE MODROB Grant (F. No. 9-93/IDC/MODROB/POLICY-1/2019-20) at Pillai College of Engineering, New Panvel.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Khedkar Aboli	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Uday Pandit Khot	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Balaji G. Hogade												✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are openly available in [https://github.com/peterhao89/A51Attacks] at doi: https://doi.org/10.1049/ise2.12120, reference number [22].

REFERENCES

[1] M. Madani and C. Tanougast, "FPGA implementation of an optimized A5/3 encryption algorithm," *Microprocessors and Microsystems*, vol. 78, p. 103212, Oct. 2020, doi: 10.1016/j.micpro.2020.103212.
[2] E. Biham and O. Dunkelman, "Cryptanalysis of the A5/1 GSM stream cipher," in *International Conference on Cryptology*, Springer Berlin Heidelberg, 2000, pp. 43–51.
[3] J. Shah and A. Mahalanobis, "A new Guess-and-determine attack on the A5/1 stream cipher," *arXiv:1204.4535*, 2012.
[4] A. Maximov, T. Johansson, and S. Babbage, "An improved correlation attack on A5/1," in *International Workshop on Selected Areas in Cryptography*, Springer Berlin Heidelberg, 2004, pp. 1–18.
[5] Z. Li, "Optimization of rainbow tables for practically cracking GSM A5/1 based on validated success rate modelling," in *Cryptographers' Track at the RSA Conference*, Springer International Publishing, 2016, pp. 359–377.
[6] T. Gendrullis, M. Novotný, and A. Rupp, "A real-world attack breaking A5/1 within hours," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 266–282.
[7] E. Barkan and E. Biham, "Conditional estimators: an effective attack on A5/1," in *International Workshop on Selected Areas in Cryptography*, Springer Berlin Heidelberg, 2006, pp. 1–19.
[8] B. Zhang, "Cryptanalysis of GSM encryption in 2G/3G networks without rainbow tables," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2019, pp. 428–456, doi: 10.1007/978-3-030-34618-8_15.
[9] A. Semenov, I. Otpuschennikov, I. Gribanova, O. Zaikin, and S. Kochemazov, "Translation of algorithmic descriptions of discrete functions to SAT with applications to cryptanalysis problems," *Logical Methods in Computer Science* , vol. 16, no. 1, 2020, doi: 10.23638/LMCS-16(1:29)2020.
[10] D. Lee, D. Hong, J. Sung, S. Kim, and S. Hong, "Improved ciphertext-only attack on GMR-1," *IEEE Access*, vol. 10, pp. 1979–1989, 2022, doi: 10.1109/ACCESS.2021.3139614.
[11] G. Avoine, X. Carpent, T. Claverie, C. Devine, and D. Leblanc-Albarel, "Time-memory trade-offs sound the death knell for GPRS and GSM," in *Annual International Cryptology Conference*, 2024, pp. 206–240, doi: 10.1007/978-3-031-68385-5_7.
[12] P. K. Gundaram, A. N. Tentu, and S. N. Allu, "State transition analysis of GSM encryption algorithm A5/1," *Journal of Communications Software and Systems*, vol. 18, no. 1, pp. 36–41, 2022, doi: 10.24138/jcomss-2021-0104.
[13] S. N. Allu and A. N. Tentu, "Quantum cryptanalysis on A5/1 Stream cipher," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 14, pp. 128–137, 2022.
[14] P. K. Gundaram, S. N. Allu, N. Yerukala, and A. N. Tentu, "Rainbow tables for cryptanalysis of A5/1 stream cipher," in *Second International Conference on Networks and Advances in Computational Technologies: NetACT 19*, 2021, pp. 251–261, doi: 10.1007/978-3-030-49500-8_22.
[15] A. Biryukov, A. Shamir, and D. Wagner, "Real time cryptanalysis of A5/1 on a PC," in *International Workshop on Fast Software*

Encryption, Springer Berlin Heidelberg, 2001, pp. 1–18.
[16] T. Pornin and J. Stern, "Software-hardware trade-offs: application to A5/1 cryptanalysis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1965 LNCS, pp. 318–327, 2000, doi: 10.1007/3-540-44499-8_25.
[17] J. Lu, Z. Li, and M. Henricksen, "Time-memory trade-off attack on the GSM A5/1 stream cipher using commodity GPGPU," in *Int. Conference on Applied Cryptography and Network Security*, 2015, pp. 350–369, doi: 10.1007/978-3-319-28166-7_17.
[18] L. Veronese, F. Palmarini, R. Focardi, and F. Luccio, "A fast and cost-effective design for FPGA-based fuzzy rainbow tradeoffs," in *Proceedings of the 8th International Conference on Information Systems Security and Privacy*, 2022, pp. 165–176, doi: 10.5220/0010904300003120.
[19] H. Hadipour and M. Eichlseder, "Autoguess: A tool for finding Guess-and-determine attacks and key bridges," in *International Conference on Applied Cryptography and Network Security*, 2022, pp. 230–250, doi: 10.1007/978-3-031-09234-3_12.
[20] A. Jain, I. Kaur, A. K. Sharma, N. K. Gupta, and P. Chakraborty, "A new Guess-and-determine method for cryptanalysis of the GSM encryption," *Complexity*, vol. 2023, pp. 1–9, Feb. 2023, doi: 10.1155/2023/7249127.
[21] J. D. Golić, "Cryptanalysis of alleged A5 stream cipher," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 1997, pp. 239–255, doi: 10.1007/3-540-69053-0_17.
[22] Y. Xu, Y. Hao, and M. Wang, "Revisit two memoryless state-recovery cryptanalysis methods on A5/1," *IET Information Security*, vol. 17, no. 4, pp. 626–638, Jul. 2023, doi: 10.1049/ise2.12120.
[23] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and Y. Papaefstathiou, "A survey of lightweight stream ciphers for embedded systems," *Security and Communication Networks*, vol. 9, no. 10, pp. 1226–1246, 2016, doi: 10.1002/sec.1399.
[24] S. B. Sadkhan and Z. Hamza, "Proposed enhancement of A5/1 stream cipher," in *2019 2nd International Conference on Engineering Technology and its Applications, IICETA 2019*, 2019, pp. 111–116, doi: 10.1109/IICETA47481.2019.9013008.
[25] P. Derbez, P.-A. Fouque, and V. Mollimard, "Fake near collisions attacks," in *IACR Transactions on Symmetric Cryptology*, Dec. 2020, pp. 88–103, doi: 10.46586/tosc.v2020.i4.88-103.

BIOGRAPHIES OF AUTHORS

Khedkar Aboli Audumbar obtained her M.E. (electronics engineering) from Mumbai University, India, in 2016, B.Tech. (Electronics and Telecommunication Engineering) from SGGS, Nanded, India, in 2013. Currently pursuring Ph.D. (electronics engineering) from Mumbai University, India. Currently, she is assistant professor in Pillai College of Engineering, New Panvel, India. She has supervised six undergraduate projects and theses. Her field of interest are embeded systems, SQL, data analytics, and cryptanalysis. She can be contacted at email:aboli@mes.ac.in.

Uday Pandit Khot obtained his B.E. (Ind. Electronics), M.Tech. (Electronic Systems) from IIT Bombay and Ph.D. (by research in the topic synthesis of analog circuits employing current-mode building blocks) from IIT Bombay. He is having 33 years of teaching experience and is a fellow member of IETE (India), member of ISTE (India), member of IEEE (USA) and member of IE (India). He has published more than 60 research papers in National/International journals and conferences. His areas of research interest are: synthesis of analog current-mode circuits, fault diagnosis in analog and digital circuits, microwave circuits, and wireless communication systems. He can be contacted at email: udaypanditkhot@sfit.ac.in.

Balaji G. Hogade received his Ph.D. in electronics and telecommunication engineering (with specialization in smart antennas for wideband wireless networks) from NMIMS University, Mumbai in 2014. He completed his M.E. in Power Electronics from Gulbarga University, Karnataka in 1999, and his B.E. in Electronics from Marathwada University, Aurangabad in 1991. Currently, he is a professor and head of the Department of Electronics Engineering at Terna Engineering College, affiliated with the University of Mumbai. He has supervised numerous undergraduate and postgraduate projects and theses. Under his guidance, five research scholars have completed their Ph.D. degrees and one has submitted the thesis from the University of Mumbai. He also serves as a Ph.D. reviewer and examiner for Savitribai Phule Pune University. He has a strong research portfolio, with several national and international publications to his credit. He has also published four patents. He was a reviewer for Various of International conferences. He has given expert talks in international conferences. Dr. Hogade was a member of the board of studies (BOS) in Electronics Engineering and Electrical Engineering at the University of Mumbai. His primary research interests include wireless networks, smart antennas, and power electronics & drives. He can be contacted at email: balajihogade@ternaengg.ac.in.