A Low Data Complexity Attack on the GMR-2 Cipher Used in the Satellite Phones

Ruilin Li[1(✉)], Heng Li[2], Chao Li[2], and Bing Sun[2]

[1] College of Electronic Science and Engineering,
National University of Defense Technology, Changsha 410073,
Hunan, People's Republic of China
securitylrl@gmail.com
[2] College of Science, National University of Defence Technology,
Changsha 410073, Hunan, People's Republic of China
lihlsm1989@gmail.com,
lichao_nudt@sina.com,
happy_come@163.com

Abstract. The GMR-1 and GMR-2 stream ciphers, which are used in the satellite phones, have been reconstructed by Driessen et al. recently. The GMR-1 cipher is shown to be a proprietary variant of the GSM A5/2 algorithm, thus it could be cracked using the previous known method. For the newly designed GMR-2 cipher, by observing a non-uniform behavior of its component, Driessen et al. proposed an efficient known plaintext attack to recover the encryption key (a session key with 64-bit) with approximately 5–6 frames (50–65 bytes) of keystream.

In this paper, we first revisit the properties of each component of the GMR-2 cipher, and then present a low data complexity attack on it by adopting the strategy of guess-and-determine. We call this kind of attack the *dynamic guess and determine attack*, since the evolution of the guessing part of the internal state of the attack is changed dynamically according to the intermediate process. Our theoretical analysis demonstrates that, using the proposed attack, the 64-bit encryption key could be recovered by guessing no more than 32 bits when 15 bytes (1 frame) of the keystream is available. Some experimental results are also performed on a single PC to confirm our analysis, and the number of candidates for exhaustive search is about 2^{28} on average.

Keywords: Satellite phone · Stream cipher · GMR-2 · Guess-and-determine

1 Introduction

1.1 Backgrounds and the GMR-2 Cipher

Nowadays, mobile communication systems have revolutionized the way we interact with each other, and there have been built many cellular mobile network such

S. Moriai (Ed.): FSE 2013, LNCS 8424, pp. 485–501, 2014.
DOI: 10.1007/978-3-662-43933-3_25,

as GSM, UMTS, CDMA2000, or 3GPP LTE. These cellular mobile networks all require a so called cell site to create a cell within the network, which provides all the necessary equipment for transmitting and receiving radio signals from mobile handsets and the radio network. However, in some cases, such as the crew on oil rig or ships on open sea, researchers on a field trip in a desert, or people living in remote areas or areas that are affected by a natural disaster, it is not always to be close to a mobile phone cellular network, then these residents, military and government systems need to use satellite phones to communicate.

Satellite phone is a type of mobile phone that connects to orbiting satellites instead of terrestrial cell sites. They provide similar functionality to terrestrial mobile telephones such as the voice, short messaging service etc. Currently, there are two major satellite phone standards both developed by ETSI, namely the GMR-1 standard and the GMR-2 (aka GMR-2+) standard. For instance, Thuraya phone implements the GMR-1 standard, while the GMR-2 standard is mainly used by Inmarsat[1].

As we all know, security plays a significant role for satellite phones, yet from ETSI, we can only obtain the specifications of those two standards without any information about implementation details of security aspects. In fact, these two standards employ two different encryption algorithms called GMR-1 cipher and GMR-2 cipher, whose details had not been publicly known until [7] was reported in January 2012.

According to [7], the GMR-1 cipher is an improved version of A5/2 which belongs to the GSM encryption standard. Thus the methods of analyzing A5/2 as introduced in [3,5] can almost be applied to the GMR-1 cipher [8]. The GMR-2 cipher is a newly designed stream cipher, and at present, only [7] presents a known plaintext attack against GMR-2 cipher which is based on the read-collision technique. This method needs approximately 50–65 bytes (5–6 frames) of the keystream to recover the full key, and the computational complexity is about 2^{18}.

1.2 Main Contribution and the Outline

In this paper, we propose a low data complexity attack on the GMR-2 cipher using the guess and determine approach. Guess-and-determine attack is a common cryptanalytic approach against stream ciphers [1,2,4,6,9–12,15]. Its basic idea is to guess some parts of the internal state and derive other part through the relationship between the keystream and the internal state introduced by the keystream generation process. The validity of a guessed and determined internal state is checked by running the cipher forward from that state. If the generated keystream matches the intercepted keystream, we accept it. Otherwise, we discard the current candidate and try the attack again.

[1] Recently, the work in [13] shows that they can modify the firmware of a Inmarsat IsatPhonePro satellite phone using only a USB cable, which allows to read and write frames directly to any layer of the GMR-2 communication system, or even allows users to inject and/or sniff frames without the need of any additional equipment.

The general guess-and-determine attack assumes that the guessed part and the corresponding determined part of the internal state are known to the adversary prior to mounting the attack procedure. However, this approach cannot directly applied to the GMR-2 cipher due to its special structure. Considering this, we present a new strategy for guess-and-determine attack which we call *the dynamic guess-and-determine*. In this strategy, the evolution of guessing part of the internal state is changed dynamically according to the intermediate process, i.e., the new guessing part depends on both the previous guessed and determined parts of the internal state. We show how this kind of attack can be used to analyze the GMR-2 stream cipher. Our theoretical analysis demonstrates that, using the proposed attack, the 64-bit session key could be recovered by guessing no more than 32 bits when 15 bytes (1 frame) of the keystream are available. The experimental results also confirm our analysis, and the number of candidates for exhaustive search is about 2^{28} on average.

The rest of this paper is organized as follows: In Sect. 2, we recall the GMR-2 cipher briefly. Section 3 gives some properties of the components of the cipher and Sect. 4 gives basic analysis of the cipher. Section 5 presents our low data complexity attack on GMR-2 cipher in detail and finally Sect. 6 concludes this paper.

2 Description of the GMR-2 Cipher

2.1 Overall Structure of the GMR-2 Cipher

The GMR-2 cipher uses a 64-bit encryption-key, denoted as $K = \{K_7, K_6, \cdots, K_0\}$ and operates on bytes. When the cipher is clocked, it generates one byte of the keystream denoted by Z_l, where l represents the number of clockings. The cipher exhibits an 8-byte state register $S = (S_7, S_6, \cdots, S_0)$, three major components $\mathcal{F}, \mathcal{G}, \mathcal{H}$, a 3-bit counter $c \in \{0, 1, \cdots, 7\}$ and a toggle-bit $t \in \{0, 1\}$. A schematic overview of the overall structure is depicted in Fig. 1.

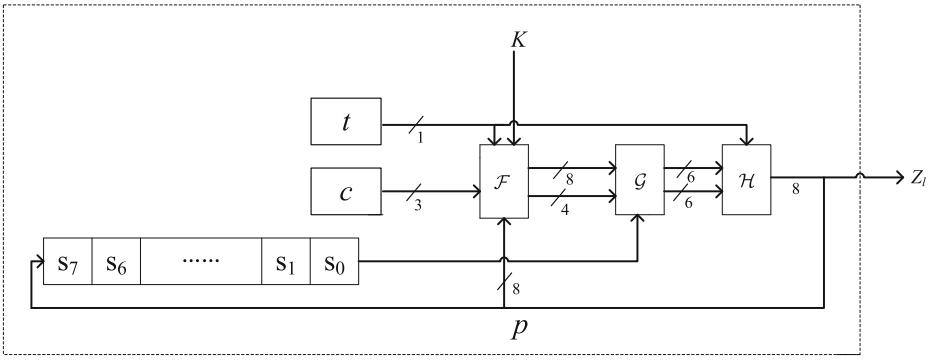

Fig. 1. Overall structure of the GMR-2 cipher

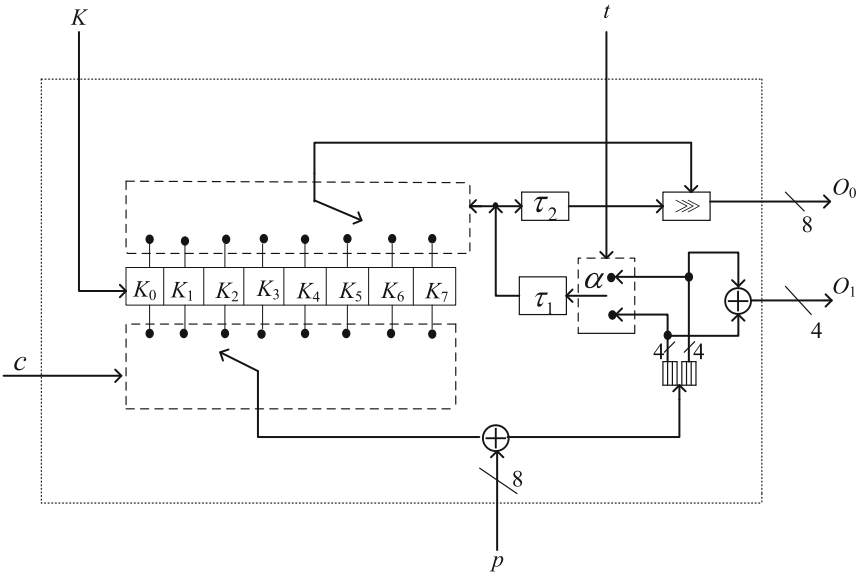

Fig. 2. The structure of $\mathcal{F}$-component

The $\mathcal{F}$-component combines two bytes of the encryption-key with the previous output (a keystream byte), the $\mathcal{G}$-component is a linear function for mixing purpose, and the $\mathcal{H}$-component consists of two DES S-boxes as a nonlinear filter. In the following subsections, we will describe the three major components in detail.

2.2 $\mathcal{F}$-Component

The $\mathcal{F}$-component is the most interesting part of the cipher, and Fig. 2 shows its internal structure. The 64-bit encryption-key K=(K_7, K_6, $\cdots$, K_0) is fed into a 64-bit resister and it is unchanged during the execution of the cipher. At each clock, the $\mathcal{F}$-component just selects two key bytes K_c and $K_{\tau_1(\alpha)}$ from the lower side and the upper side, which can be described formally as follows.

Assume the cipher is at the l-th clock, besides the encryption-key K, the inputs of the $\mathcal{F}$-component contain t, c and p, where $c = l \mod 8$ is a counter ranging from 0 to 7 sequentially and repeatedly, $t = c \mod 2$ is a toggle bit, and $p = (p_7, p_6, \cdots, p_0) \in \{0,1\}^8$ is one byte of the keystream that has already been generated in the last clock. We will simply use $p = Z_{l-1}$ to denote one byte of the keystream that has already been generated. The outputs of $\mathcal{F}$-component contain an 8-bit O_0 and a 4-bit O_1 of the following form

$$\begin{cases} O_0 = (K_{\tau_1(\alpha)} \ggg \tau_2(\tau_1(\alpha)))_8; \\ O_1 = ((((K_c \oplus p) \gg 4) \& \mathsf{0xF}) \oplus ((K_c \oplus p) \& \mathsf{0xF}))_4. \end{cases} \tag{1}$$

Table 1. Definition of τ_1 and τ_2

α	$\tau_1(\alpha)$	$\tau_2(\tau_1(\alpha))$	α	$\tau_1(\alpha)$	$\tau_2(\tau_1(\alpha))$
(0,0,0,0)	2	6	(1,0,0,0)	3	7
(0,0,0,1)	5	3	(1,0,0,1)	**0**	**4**
(0,0,1,0)	**0**	**4**	(1,0,1,0)	6	2
(0,0,1,1)	6	2	(1,0,1,1)	1	5
(0,1,0,0)	3	7	(1,1,0,0)	5	3
(0,1,0,1)	7	1	(1,1,0,1)	7	1
(0,1,1,0)	**4**	**4**	(1,1,1,0)	**4**	**4**
(0,1,1,1)	1	5	(1,1,1,1)	2	6

where $\tau_1 : \{0,1\}^4 \longrightarrow \{0,1\}^3$ and $\tau_2 : \{0,1\}^3 \longrightarrow \{0,1\}^3$ are defined by table-lookups as shown in Table 1, and α is defined as

$$\alpha = \mathcal{N}(t, K_c \oplus P) = \begin{cases} ((K_c \oplus p)\&\mathsf{0xF}))_4, & \text{if} \quad t = 0; \\ (((K_c \oplus p) \gg 4)\&\mathsf{0xF})_4, & \text{if} \quad t = 1, \end{cases} \tag{2}$$

which can also be expressed using the following simple form

$$\alpha = [(K_c \oplus p) \gg 4 \times (c \bmod 2)] \,\&\, \mathsf{0xF}.$$

2.3 $\mathcal{G}$-Component

As demonstrated in Fig. 3, the $\mathcal{G}$-component gets the output of the $\mathcal{F}$-component and one byte S_0 of the state as its input. It employs three sub-components, denoted by $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, all work on 4-bit input and returns 4-bit output with the

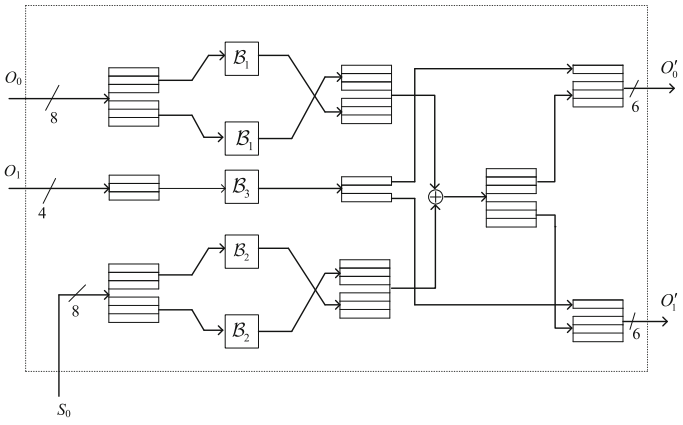

Fig. 3. The structure of $\mathcal{G}$-component (the upper lines indicates lower bits)

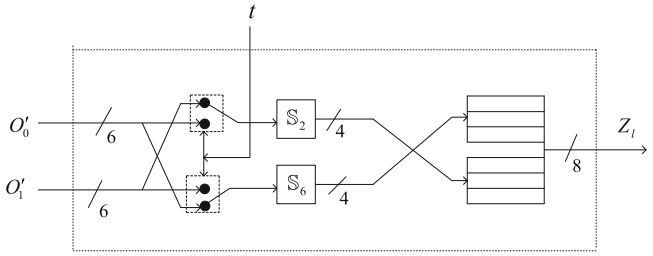

Fig. 4. The structure of $\mathcal{H}$-component

following definitions

$$\begin{cases} \mathcal{B}_1 : (x_3, x_2, x_1, x_0) \mapsto (x_3 \oplus x_0, x_3 \oplus x_2 \oplus x_0, x_3, x_1); \\ \mathcal{B}_2 : (x_3, x_2, x_1, x_0) \mapsto (x_1, x_3, x_0, x_2); \\ \mathcal{B}_3 : (x_3, x_2, x_1, x_0) \mapsto (x_2, x_0, x_3 \oplus x_1 \oplus x_0, x_3 \oplus x_0). \end{cases}$$

Since each $\mathcal{B}_i$ is linear, and all the other operations are just transposition or XOR, the $\mathcal{G}$-component is an entirely linear transformation, and we can express the two 6-bit outputs O'_0 and O'_1 as linear functions of the input by Eq. (3)

$$\begin{cases} O'_0 = (O_{0,7} \oplus O_{0,4} \oplus S_{0,5},\ O_{0,7} \oplus O_{0,6} \oplus O_{0,4} \oplus S_{0,7},\ O_{0,7} \oplus S_{0,4}, \\ \qquad O_{0,5} \oplus S_{0,6},\ O_{1,3} \oplus O_{1,1} \oplus O_{1,0},\ O_{1,3} \oplus O_{1,0}) \\ O'_1 = (O_{0,3} \oplus O_{0,0} \oplus S_{0,1},\ O_{0,3} \oplus O_{0,2} \oplus O_{0,0} \oplus S_{0,3},\ O_{0,3} \oplus S_{0,0}, \\ \qquad O_{0,1} \oplus S_{0,2},\ O_{1,2},\ O_{1,0}). \end{cases} \tag{3}$$

2.4 $\mathcal{H}$-Component

The input of the $\mathcal{H}$-component as shown in Fig. 4, is the outputs of $\mathcal{G}$-component O'_0, O'_1 and a toggle-bit t.

$\mathcal{H}$-component contains two S-boxes $\mathbb{S}_2$ and $\mathbb{S}_6$, where $\mathbb{S}_2$ is the second S-box of DES and $\mathbb{S}_6$ is the sixth S-box of DES. See Tables 2 and 3 for a reference. However, these two S-boxes have been reordered to account for the different addressing.

Assume the input of S-box is $(x_6, x_5, x_4, x_3, x_2, x_1)$, then in this cipher, the least-significant bits (x_2, x_1) select the S-box row and the four most-significant

Table 2. The S-box $\mathbb{S}_2$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Table 3. The S-box $\mathbb{S}_6$

	0	1	2	3	**4**	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	**9**	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	**9**	5	15	10	11	14	1	7	6	0	8	13

bits (x_6, x_5, x_4, x_3) select the S-box column. Now depending on the value of t, the output of $\mathcal{H}$-component, which is the l-th byte of the keystream, can be defined by

$$Z_l = \begin{cases} (\mathbb{S}_2(O'_1), \mathbb{S}_6(O'_0))_8, & \text{if} \quad t = 0; \\ (\mathbb{S}_2(O'_0), \mathbb{S}_6(O'_1))_8, & \text{if} \quad t = 1. \end{cases} \tag{4}$$

2.5 Mode of Operation

Now, we can describe the mode of operation [7] for the GMR-2 cipher. When the cipher is clocked for the l-th time, the following happens:

- Based on the current state of the state-register S, the counter c, and the toggle-bit t, the cipher generates one byte Z_l of keystream.
- The counter c is incremented by one and the toggle-bit is computed as $t = c \bmod 2$. When 8 is reached for c, then c is reset to 0.
- The state-register S is shifted by 8 bits to the right: $S_i = S_{i+1}$, $i = 0, 1, \ldots, 6$, and $S_7 = Z_l$. Meanwhile, $p = Z_l$ is also passed to the $\mathcal{F}$-component as input for the next iteration (the $(l+1)$-th clock).

The cipher is operated in two modes, the initialization mode and the generation mode.

Initialization Mode. In the initialization phase, the following steps are performed:

- The counter $c = 0$ and the toggle-bit $t = 0$.
- The 64-bit encryption-key is written into the resister in the $\mathcal{F}$-component.
- The state-register S is initialized with the 22-bit frame-number N, and this procedure is not detailed here as it is irrelevant with our attack. After c, t, S have been initialized, the cipher is clocked eight times, but the resulting keystream is *discarded*.

Generation Mode.[2] After the initialization is finished, the cipher is clocked to generate and output actual keystream bytes. We use $Z_l^{(N)}$ to denote the l-th $(l \geq 0)$ byte of keystream generated after initialization with frame-number N.

[2] There is a slight difference between the notation of [7] and ours in the generation mode, in this paper, we always assume that $Z_0^{(N)}$ is the first output byte of the keystream after the cipher is clocked eight times in the initialization phase.

The frame-number is always incremented after 15 bytes of keystream, which forces a re-initialization of the cipher. Therefore the keystream Z' that is actually used for $N \in \{0, 1, \cdots\}$ is made up of blocks of 15 bytes that are concatenated as follows:

$$Z' = (Z_0^{(0)}, Z_1^{(0)}, \cdots, Z_{14}^{(0)}, Z_0^{(1)}, Z_1^{(1)}, \cdots, Z_{14}^{(1)}, \cdots).$$

3 Properties of the Components of the GMR-2 Cipher

In this section, we carefully study the characteristic of the GMR-2 cipher and propose several properties of its components which are related to our later analysis.

3.1 Property of the $\mathcal{F}$-Component

We first note that after the 64-bit encryption key K is fed into the $\mathcal{F}$-component, it remains unchanged not only in the phase of the initialization, but also in the phrase of the keystream generation. Since the $\mathcal{F}$-component is used to select two key bytes K_c and $K_{\tau_1(\alpha)}$ from K, and the counter c is changed sequentially from 0–7, we only need to know how $K_{\tau_1(\alpha)}$ is selected.

Property of α. By Eq. (2), α can be expressed as:

$$\begin{aligned}\alpha &= \mathcal{N}(t, K_c \oplus p) \\ &= \begin{cases} (K_{c,3} \oplus p_3,\ K_{c,2} \oplus p_2,\ K_{c,1} \oplus p_1,\ K_{c,0} \oplus p_0)_4, & \text{if} \quad t = 0; \\ (K_{c,7} \oplus p_7,\ K_{c,6} \oplus p_6,\ K_{c,5} \oplus p_5,\ K_{c,4} \oplus p_4)_4, & \text{if} \quad t = 1. \end{cases}\end{aligned}$$

This tells us that if p is known, then at each clock, we can get the value of α only by the four least-significant bits of K_c when $t = 0$ (c is even) or the four most-significant bits of K_c when $t = 1$ (c is odd). Thus, the key byte $K_{\tau_1(\alpha)}$ selected by the upper side can be determined by the value of the most (least) significant 4-bit of K_c provided p is known.

Properties of τ_1 *and* τ_2. From Table 1, we know that τ_1 maps 4-bit to 3-bit, thus a collision always exists. For instance, $\tau_1(0, 0, 1, 0) = \tau_1(1, 0, 0, 1) = 0$, and $\tau_1(0, 1, 1, 0) = \tau_1(1, 1, 1, 0) = 4$, this observation combined with $\tau_2(0) = \tau_2(4) = 4$ lead to the efficient read-collision based attack in [7]. Note that $\tau_2(\cdot)$ maps 3-bit to 3-bit, but it is non-surjective. Since one of the output of $\mathcal{F}$-component is $O_0 = K_{\tau_1(\alpha)} \ggg \tau_2(\tau_1(\alpha))$, we guess the reason why the designers choose a non-surjective table for $\tau_2(\cdot)$, he just want to make the right rotation parameter always being non-zero. Currently, we do not know whether this kind of non-uniformity could lead to some other potential attacks.

3.2 Property of the $\mathcal{G}$-Component

According to Eq. (3), the link between the input and output of the $\mathcal{G}$-component can be expressed by

$$\begin{pmatrix} O'_{0,5} \\ O'_{0,4} \\ O'_{0,3} \\ O'_{0,2} \\ O'_{1,5} \\ O'_{1,4} \\ O'_{1,3} \\ O'_{1,2} \\ O'_{0,1} \\ O'_{0,0} \\ O'_{1,1} \\ O'_{1,0} \end{pmatrix} = \begin{pmatrix} 1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1 \end{pmatrix} \cdot \begin{pmatrix} O_{0,7} \\ O_{0,6} \\ O_{0,5} \\ O_{0,4} \\ O_{0,3} \\ O_{0,2} \\ O_{0,1} \\ O_{0,0} \\ O_{1,3} \\ O_{1,2} \\ O_{1,1} \\ O_{1,0} \end{pmatrix} \oplus \begin{pmatrix} S_{0,5} \\ S_{0,7} \\ S_{0,4} \\ S_{0,6} \\ S_{0,1} \\ S_{0,3} \\ S_{0,0} \\ S_{0,2} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{5}$$

based on which we can obtain the following three linear equation systems:

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{x} \oplus \mathbf{v}, \tag{6}$$

$$\mathbf{y_1} = \mathbf{W_1} \cdot \mathbf{x_1} \oplus \mathbf{v_1}, \tag{7}$$

$$\mathbf{y_2} = \mathbf{W_2} \cdot \mathbf{x_2} \oplus \mathbf{v_2}, \tag{8}$$

where

$$\mathbf{W} = \begin{pmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B} \end{pmatrix}, \quad \mathbf{W_1} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{pmatrix}, \quad \mathbf{W_2} = \begin{pmatrix} \mathbf{B} \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} 1\,0\,0\,1 \\ 1\,1\,0\,1 \\ 1\,0\,0\,0 \\ 0\,0\,1\,0 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 1\,0\,1\,1 \\ 1\,0\,0\,1 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \end{pmatrix}, \quad \mathbf{0} = \begin{pmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{pmatrix},$$

and

$$\begin{cases} \mathbf{y_1} = (O'_{0,5}, O'_{0,4}, O'_{0,3}, O'_{0,2}, O'_{1,5}, O'_{1,4}, O'_{1,3}, O'_{1,2})^T \\ \mathbf{y_2} = (O'_{0,1}, O'_{0,0}, O'_{1,1}, O'_{1,0})^T \\ \mathbf{y} \ \ = (\mathbf{y_1}, \mathbf{y_2}) \end{cases},$$

$$\begin{cases} \mathbf{x_1} = (O_{0,7}, O_{0,6}, O_{0,5}, O_{0,4}, O_{0,3}, O_{0,2}, O_{0,1}, O_{0,0})^T \\ \mathbf{x_2} = (O_{1,3}, O_{1,2}, O_{1,1}, O_{1,0})^T \\ \mathbf{x} \ \ = (\mathbf{x_1}, \mathbf{x_2}) \end{cases},$$

$$\begin{cases} \mathbf{v_1} = (S_{0,5}, S_{0,7}, S_{0,4}, S_{0,6}, S_{0,1}, S_{0,3}, S_{0,0}, S_{0,2})^T \\ \mathbf{v_2} = (0, 0, 0, 0)^T \\ \mathbf{v} \ \ = (\mathbf{v_1}, \mathbf{v_2}) \end{cases}.$$

Further, let $K_c = (\mathbf{k_h}, \mathbf{k_l})$, where $\mathbf{k_h} = (K_{c,7}, K_{c,6}, K_{c,5}, K_{c,4})^T$ denotes the most significant 4-bit of K_c, and $\mathbf{k_l} = (K_{c,3}, K_{c,2}, K_{c,1}, K_{c,0})^T$ denotes the least

significant 4-bit of K_c. Similarly let $p = (\mathbf{p_h}, \mathbf{p_l})$, where $\mathbf{p_h} = (p_7, p_6, p_5, p_4)^T$, $\mathbf{p_l} = (p_3, p_2, p_1, p_0)^T$, and define $\mathbf{u} = \mathbf{p_h} \oplus \mathbf{p_l}$, then Eq. (1) implies the following two linear systems

$$\mathbf{x_1} = K_{\tau_1(\alpha)} \ggg \tau_2(\tau_1(\alpha)) \tag{9}$$

$$\mathbf{x_2} = \mathbf{k_h} \oplus \mathbf{k_l} \oplus \mathbf{u} \tag{10}$$

In the following attack on the GMR-2 cipher, we will always use one of the above linear systems, and we can guarantee that both the exact values of $\mathbf{u}$ and $\mathbf{v}$ are known to us. We have the following observations:

Observation 1. Since $\mathbf{A}$ and $\mathbf{B}$ are invertible, so are $\mathbf{W}$, $\mathbf{W_1}$ and $\mathbf{W_2}$, then from Eqs. (6)–(8), we can obtain the value of $\mathbf{y}$ ($\mathbf{y_i}$) from $\mathbf{x}$ ($\mathbf{x_i}$) easily, and vice vera.

Observation 2. If both $\mathbf{y_1}$ and α are known, then from observation 1, we can get the value of $\mathbf{x_1}$, and further from Eq. (9), we can calculate $K_{\tau_1(\alpha)} = \mathbf{x_1} \lll \tau_2(\tau_1(\alpha))$.

Observation 3. If both $\mathbf{y_2}$ and $\mathbf{k_h}$ ($\mathbf{k_l}$) are known, then from observation 1, we can get the value of $\mathbf{x_2}$, and further from Eq. (10), we can calculate $\mathbf{k_l} = \mathbf{x_2} \oplus \mathbf{k_h} \oplus \mathbf{u}$ ($\mathbf{k_h} = \mathbf{x_2} \oplus \mathbf{k_l} \oplus \mathbf{u}$).

Observation 4. The column indices of the two S-boxes $\mathbb{S}_2$ and $\mathbb{S}_6$ are selected by $\mathbf{y_1}$, and the row indices are selected by $\mathbf{y_2}$. This relationship is depicted in Fig. 5.

3.3 Property of the $\mathcal{H}$-Component

According to Eq. (4) and the definition of the two S-boxes, we have the following three results:

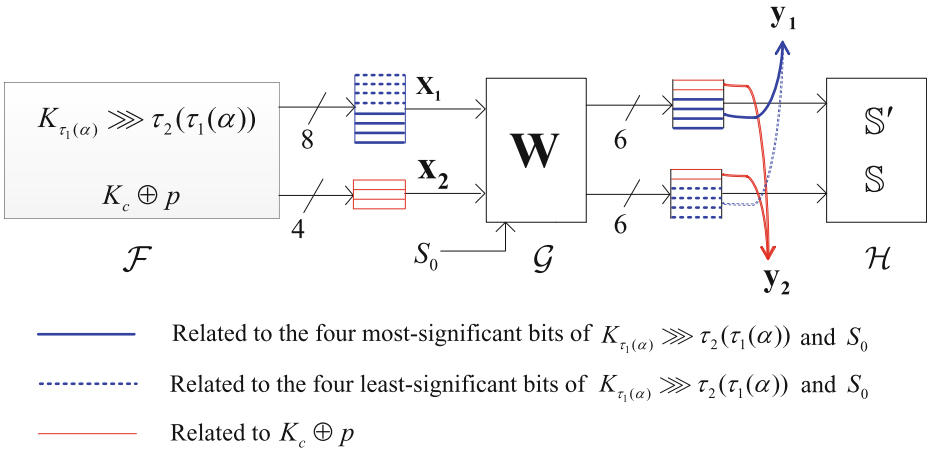

Fig. 5. The links between the input and output of the $\mathcal{G}$-component (the upper lines indicates lower bits). Note that $\alpha = [(K_c \oplus p) \gg 4 \times (c \mod 2)]$ & 0xF

– If the row index and the output of an S-box are known, then we will get the column index uniquely.
– If the column index and the output of an S-box are known, then we will also get the row index uniquely except for $\mathbb{S}_6$ when the column index is 4 and the output is 9, in this situation, the row index can be either 0 or 3.
– If only the outputs of both S-boxes are known, then we will get $4 \times 4 = 16$ possible inputs for $\mathcal{H}$-component.

The above three results indicate that by intercepting the keystream of the GMR-2 cipher (the output of the two S-boxes) and combining the guessed/determined values of the row or column indices, we can "invert" these two S-boxes to obtain the corresponding (partial) input candidates.

4 Basic Analysis of the GMR-2 Cipher

The previous section presents some properties of the three components of the GMR-2 cipher. In this section, we show how these components interact with each other.

Given the frame number N, let $S_i^{(l)}$ denote the state of S_i at the l-th ($0 \leq l \leq 14$) clock in the keystream generation phrase, then for $8 \leq l \leq 14$ we have

$$S_0^{(l)} = Z_{l-8}^{(N)} \quad \text{and} \quad p = S_7^{(l)} = Z_{l-1}^{(N)},$$

which indicates that for $8 \leq l \leq 14$, both $S_0^{(l)}$ and p are known to us, thus the vectors $\mathbf{v}$, $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{u}$ as defined in the previous section are also known to us. To simply our analysis, in the following of this section, we only focus on the cipher at the $(c+8)$-th clock with $0 \leq c \leq 6$.

Note that at the $(c+8)$-th clock, the $\mathcal{F}$-component just selects two key bytes K_c and $K_{\tau_1(\alpha)}$ from the lower side and the upper side. According to the property of the $\mathcal{F}$-component, just by guessing the half value of $K_c = (\mathbf{k_h}, \mathbf{k_l})$, we can determine the value of α and then know which key byte the $\mathcal{F}$-component will select.

Now, based on the fact that the link between the input and output of the $\mathcal{G}$-component can be expressed by a well-structured matrix $\mathbf{W}$, we present the following four rules for the guessing strategy when applying the dynamic guess-and-determine attack as described in the next section.

Rule 1. Let $K_c = (\mathbf{k_h}, \mathbf{k_l})$, assume c is odd, and given a guessed value for $\mathbf{k_h}$, if $\tau_1(\alpha) = c$, then from $Z_{c+8}^{(N)}$, either the guessed value of $\mathbf{k_h}$ is wrong or the candidate value of $\mathbf{k_l}$ can be determined; Similarly, assume c is even, and given a guessed value for $\mathbf{k_l}$, if $\tau_1(\alpha) = c$, then from $Z_{c+8}^{(N)}$, either the guessed value of $\mathbf{k_l}$ is wrong or the candidate value of $\mathbf{k_h}$ can be determined.

Proof. We only give the proof for the first case, the other case is similar, and thus the detail is omitted.

From $\tau_1(\alpha) = c$, we have $K_{\tau_1(\alpha)} = K_c$, thus

$$\mathbf{x_1} = K_{\tau_1(\alpha)} \ggg \tau_2(\tau_1(\alpha)) = (\mathbf{k_h}, \mathbf{k_l}) \ggg \tau_2(\tau_1(\alpha)).$$

Noting that

$$\mathbf{x_2} = \mathbf{k_h} \oplus \mathbf{k_l} \oplus \mathbf{u} \quad \text{and} \quad \mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}),$$

thus if $\mathbf{k_h}$ is known, then for each possible $\mathbf{y}$ (whose value is calculated later), Eq. (6) can be converted into another linear equation system (which is related to the guessed value of $\mathbf{k_h}$) with 12 equations and 4 indeterminate variables representing $\mathbf{k_l}$.

According to the properties of the $\mathcal{H}$-component, there will be 16 different values for $\mathbf{y} = (\mathbf{y_1}, \mathbf{y_2})$ from $Z_{c+8}^{(N)}$. Thus, in total, 16 linear equation systems for $\mathbf{k_l}$ can be obtained.

If the guessed value of $\mathbf{k_h}$ is the actual value, at least one of the above 16 linear systems will have a solution that can be find through Gaussian elimination method. While if $\mathbf{k_h}$ is a random guessed value, then based on the theory of *Linear Consistency Test* [14], the probability that each linear equation system has solutions is no more than

$$\frac{1}{2^{12-4}} \times \left(1 + \frac{1}{2^{12+1}}\right)^4 \approx 2^{-8}.$$

Thus, the probability that the above 16 linear equation systems have solutions is upper bounded by $16 \times 2^{-8} = 2^{-4}$. In other words, this indicates that the number of candidates for $\mathbf{k_l}$ is small. □

Rule 2. Let $K_c = (\mathbf{k_h}, \mathbf{k_l})$, assume c is odd (even), and given a guessed value for $\mathbf{k_h}$ ($\mathbf{k_l}$), if $\tau_1(\alpha) \neq c$, we further guess the value of $\mathbf{k_l}$ ($\mathbf{k_h}$), in this situation, we have a guessed value for K_c, and then $K_{\tau_1(\alpha)}$ can be determined by $Z_{c+8}^{(N)}$.

Proof. Since $K_c = (\mathbf{k_h}, \mathbf{k_l})$ is known by guess, $\mathbf{x_2} = \mathbf{k_h} \oplus \mathbf{k_l} \oplus \mathbf{u}$ is known, according to observation 1, $\mathbf{y_2}$ can be calculated. By observation 4, the row indices for the two S-boxes are known, then from $Z_{c+8}^{(N)}$, the value of $\mathbf{y_1}$ which corresponds to the column indices for the two S-boxes can be uniquely determined. By observation 2, the value of $K_{\tau_1(\alpha)}$ can be obtained. □

Rule 3. Let $K_c = (\mathbf{k_h}, \mathbf{k_l})$, assume c is odd, and given guessed value for $\mathbf{k_h}$, if $K_{\tau_1(\alpha)}$ had already been guessed or determined previously, then $\mathbf{k_l}$ can be determined by $Z_{c+8}^{(N)}$; Similarly, assume c is even, and given guessed value for $\mathbf{k_l}$, if $K_{\tau_1(\alpha)}$ had already been guessed or determined previously, then $\mathbf{k_h}$ can be determined by $Z_{c+8}^{(N)}$.

Proof. Since $K_{\tau_1(\alpha)}$ is known, $\mathbf{x_1}$ is known, by observation 1, the value of $\mathbf{y_1}$ can be obtained. Noting that $\mathbf{y_1}$ corresponds to the column indices for S-boxes, thus $\mathbf{y_2}$ which represents the row indices for S-boxes can be obtained from $Z_{c+8}^{(N)}$. According to observation 3, $\mathbf{k_h}$ ($\mathbf{k_l}$) can be calculated with known $\mathbf{k_l}$ ($\mathbf{k_h}$). □

Remark 1. We remind here that, from $\mathbf{y_1}$ and $Z_{c+8}^{(N)}$, we cannot always uniquely deduce $\mathbf{y_2}$ as explained in Sect. 3.3, thus we will sometimes obtain two candidates for $\mathbf{y_2}$.

Rule 4. Assume that the values for K_c and $K_{\tau_1(\alpha)}$ had already been guessed or determined previously, then we can judge whether those guessed or determined values are wrong by $Z_{c+8}^{(N)}$.

Proof. Since K_c and $K_{\tau_1(\alpha)}$ are known, they can pass through the three components to generate a keystream byte at the $(c+8)$-th clock, then we can compare it with $Z_{c+8}^{(N)}$. If they are not matched, the guessed values for K_c and $K_{\tau_1(\alpha)}$ are wrong. □

Remark 2. When applying dynamic guess-and-determine attack on the GMR-2 cipher in the next section, in fact, at each step, we just adopt Rule 1–Rule 3 to guess or determine some parts of K, or adopt Rule 4 to verify whether the guessed or determined value is wrong. If Rule 4 indicates some inconsistency at the current clock, then the guessed value for the nearest clock is wrong, in this situation, we must *backtrack* to this position, and try another guessed value.

5 Low Data Complexity Attack on the GMR-2 Cipher

As discussed in the introduction, the general guess-and-determine attack assumes that both the guessed part and the corresponding determined part of the internal state are known to the adversary prior to mounting the attack. However, considering the mechanism of the GMR-2 cipher, we cannot directly applied the general guess-and-determine attack on it. Thus, we introduce a new strategy for guess-and-determine attack which we call *the Dynamic Guess-and-Determine*. The main feature is that we cannot decide which parts must be guessed and which parts have to be determined in prior, what we can do is just *dynamically* guessing some parts of the internal state. The idea can be further described as follows.

First, we guess some part of the internal state of the target cipher, and then according to the guessed value, we determine some other parts of the internal state through the intercepted keystream. Next, we continue to guess some new part of the internal state, but this time the guessed part depends on both the previous guessed and determined parts. Do this process until all parts of the internal state are deduced. This indicates that we need to dynamically build the candidates for K by *backtracking*.

Now we can adopt the above strategy to present a low data complexity attack on the GMR-2 cipher. Our attack only needs one frame (15-byte) of the keystream, and without loss of generality, we assume $N = 0$. The attack contains the following two major steps[3]:

[3] We point out here that although we describe our attack in two separate steps, in fact, the second step (the exhaustive search step) can be incorporated in the first step: if a candidate is obtained from the dynamic guess-and-determine phase, it can be quickly tested to decide whether it is the right key.

- In the first step, from the known keystream $Z_0^{(0)} \sim Z_{14}^{(0)}$, we adopt the dynamic guess-and-determine method to analyze the cipher at the $(c+8)$-th clock, where $0 \leq c \leq 6$, and this can reduce the candidates for the 64-bit encryption-key K from 2^{64} to no more than 2^{32}.
- In the second step, we test the candidates for K from the first step by comparing the keystream generated from these candidates with the exact keystream $Z_0^{(0)} \sim Z_7^{(0)}$, thus we obtain the unique value for K.

Since the second step of our attack is just doing exhaustive search operations for the candidate set, we only discuss the first step in detail in the following subsection.

5.1 The Attack Procedure

As explained before, to guarantee that the values of p and $S_0^{(l)}$ are known for us at the l-th clock, we should analyze the cipher at the $(c+8)$-th clock with $0 \leq c \leq 6$.

Before introducing the proposed attack, we first define an index set

$$\Gamma \subseteq \{0, 1, \cdots, 7\}$$

to save the byte indices for the encryption key K that had already been known by guessing or determining before the $(c+8)$-th clock. Γ is initialized with $\varnothing$ at the 8th clock, and is changed during the attack process.

Now let's analyze the GMR-2 cipher at the $(c+8)$-th clock with $0 \leq c \leq 6$. At each clock, we calculate the following values:

$$c,\ t,\ p = Z_{c+7}^{(0)},\ S_0^{(c+8)} = Z_c^{(0)},\ \text{and}\ \Gamma,$$

and judge whether $c \in \Gamma$:

- If $c \in \Gamma$, then K_c had been known, we could calculate α and judge whether $\tau_1(\alpha) \in \Gamma$:
  - If $\tau_1(\alpha) \in \Gamma$, then $K_{\tau_1(\alpha)}$ had been known, thus we can adopt Rule 4 to determine whether K_c and $K_{\tau_1(\alpha)}$ are wrong. If they are incorrect (i.e., the guessed and determined values are wrong), then we trace back to the nearest clock (at which the guessed value indicates such inconsistency) to re-analyze the cipher.
  - If $\tau_1(\alpha) \notin \Gamma$, then $K_{\tau_1(\alpha)}$ had not been known, we can adopt Rule 2 to obtain $K_{\tau_1(\alpha)}$, and meanwhile set $\Gamma \leftarrow \Gamma \cup \{\tau_1(\alpha)\}$.
- If $c \notin \Gamma$, then $K_c = (\mathbf{k_h}, \mathbf{k_l})$ had not been known, now we decide to guess $\mathbf{k_l}$ if c is even, and $\mathbf{k_h}$ if c is odd. Next, we calculate α and judge whether $\tau_1(\alpha) \in \Gamma$:
  - If $\tau_1(\alpha) \in \Gamma$, then $K_{\tau_1(\alpha)}$ had been known, we can adopt Rule 3 to get $\mathbf{k_h}$ if c is even, and $\mathbf{k_l}$, if c is odd, and meanwhile set $\Gamma \leftarrow \Gamma \cup \{c\}$.
  - If $\tau_1(\alpha) \notin \Gamma$, then $K_{\tau_1(\alpha)}$ had not been known. We further judge whether $c = \tau_1(\alpha)$:

If $c = \tau_1(\alpha)$, then we can adopt Rule 1 to either get the rest bits of K_c, and set $\Gamma \leftarrow \Gamma \cup \{c\}$, or deduce that the guessed value of $\mathbf{k_l}$ ($\mathbf{k_h}$) is wrong if c is even (odd), and then we guess another value for $\mathbf{k_l}$ ($\mathbf{k_h}$).
If $c \neq \tau_1(\alpha)$, then we guess the other four bits of K_c, and we can adopt Rule 2 to get $K_{\tau_1(\alpha)}$, and meanwhile set $\Gamma \leftarrow \Gamma \cup \{c, \tau_1(\alpha)\}$.

The above process sequentially executes on the GMR-2 cipher from the 8th clock to the 14th clock. When it is finished, there will be a candidate for the 64-bit K, then we test whether it is the right key by $Z_0^{(0)} \sim Z_7^{(0)}$. If not, we discard this candidate, and then we modify the guessed values to obtain another candidate. This process is repeated until the right key is found at last.

5.2 Complexity Analysis and Experimental Results

From the attack procedure, especially from Rule 1– Rule 3, it is shown that if we guess 8 bits, then we will obtain other 8 bits; while if we guess 4 bits, then we will also deduce other 4 bits. Furthermore, Rule 4 can be further used to filter the wrong guessed values. We thus conclude that for a 64-bit key K, we only need to guess at most 32 bits on average, and the other 32 bits can be determined. This estimation is rough, however, it seems difficult and even impossible to calculate the exact time complexity of our attack in theory. So we do some experiments for different frames and random keys. Our experimental results almost confirm our analysis, and the number of candidates is a little better, it is about 2^{28} on average.

More specifically, we perform a *non-optimized*[4] realization of the above attack 1000 times on a 3.2 GHz PC, and the result demonstrates that the 64-bit encryption-key can be obtained in around 700 seconds on average, where 580 seconds are consumed to deduce the 2^{28} candidates, and 120 seconds are consumed to exhaustively search the candidates. Figure 6 is the frequence distribution of the exhaustive bits (the logarithm of the number of candidates) from 1000 experimental results.

The data complexity of the attack is just a frame of the keystream, i.e., 15-byte keystream. The dynamic guess-and-determine phase only analyze 8th∼14th clock, because $S_7, S_6, \ldots, S_0$ must be known in this phase. While for the exhaustive search phase, $Z_0^{(0)} \sim Z_7^{(0)}$ can be used to distinguish the right key from the 2^{28} candidates.

[4] As described in Sect. 5.1, in the dynamic guess-and-determine attack, if we detect some inconsistency at the $(c+8)$-th clock, we should backtrack to the nearest clock. However, for easy programming with the recursive method, in our "non-optimized" realization, we just trace back to the $(c+7)$-th clock, thus there maybe exist many redundant computations. We believe that using the original realization, the time complexity of the attack can be further reduced quickly.

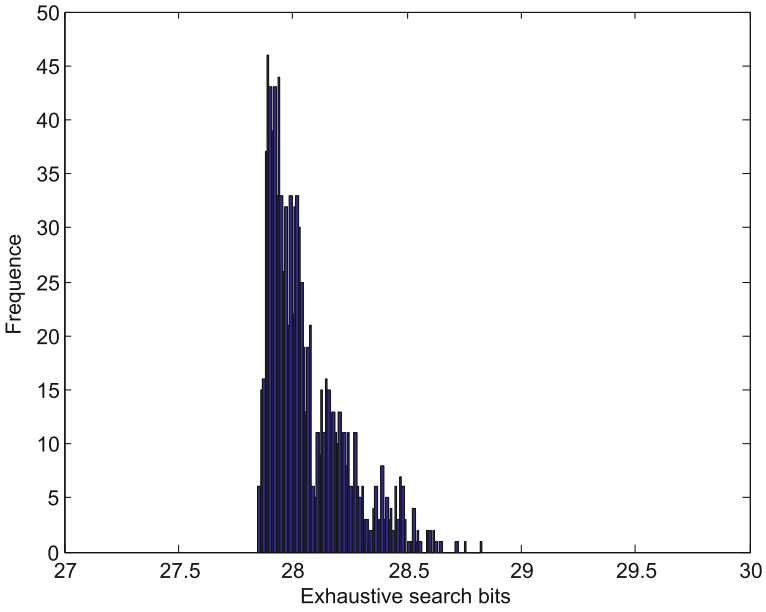

Fig. 6. The frequence distribution of exhaustive bits from 1000 experimental results

Table 4. Cryptanalytic results on the GMR-2 cipher

Method	Data	Time	Source
Read-Collision Based Technique	15–20 frames	2^{10}	[7]
Read-Collision Based Technique	5–6 frames (50–65 bytes)	2^{18}	[7]
Dynamic Guess-and-Determine	1 frames (15 bytes)	2^{28}	Sect. 5

6 Conclusion

The GMR-2 cipher has been widely used in the satellite phones communications, and thus it is of special significant to analyze its security. The design methodology of GMR-2 cipher seems new and more complex, yet an efficient low data complexity attack based on the strategy of dynamic guess-and-determine could be mounted. This kind of attack needs only 1 frame (15-byte) of the keystream, and it can recover the 64-bit session key by testing about 2^{28} candidates on average. Table 4 is the comparison between the known cryptanalytic result and ours. Our proposed attack can also be implemented on a single PC, which again demonstrates that the design methodology of the GMR-2 cipher is really far from what is "state of the art" in stream ciphers.

Acknowledgments. The authors wish to thank the anonymous reviewers of FSE 2013 for their valuable suggestions and comments, which greatly improve the presentation and quality of this paper. The work in this paper is supported by the National Natural Science Foundation of China (No: 61070215, 61103192).

References

1. Abdelraheem, M.A., Borghoff, J., Zenner, E., David, M.: Cryptanalysis of the light-weight cipher A2U2. In: Chen, L. (ed.) IMACC 2011. LNCS, vol. 7089, pp. 375–390. Springer, Heidelberg (2011)
2. Anderson R, Roe M. A5 (1994). http://jya.com/crack-a5.htm
3. Barkan, P., Biham, E., Keller, N.: Instant ciphertext-only cryptanalysis of GSM encrypted communication. J. Cryptol. (Springer) **21**(3), 392–429 (2008)
4. Biham, E., Dunkelman, O.: Cryptanalysis of the A5/1 GSM stream cipher. In: Roy, B., Okamoto, E. (eds.) INDOCRYPT 2000. LNCS, vol. 1977, pp. 43–51. Springer, Heidelberg (2000)
5. Bogdanov, A., Eisenbarth, T., Rupp, A.: A hardware-assisted realtime attack on A5/2 without precomputations. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 394–412. Springer, Heidelberg (2007)
6. Debraize, B., Goubin, L.: Guess-and-determine algebraic attack on the self-shrinking generator. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 235–252. Springer, Heidelberg (2008)
7. Driessen, B., Hund, R., Willems, C., Parr, C., Holz, T.: Don't trust satellite phones: a security analysis of two satphone standards. In: IEEE Security and Privacy 2012, pp. 128–142 (2012)
8. Driessen B. Eavesdropping on satellite telecommunication system. Cryptology ePrint Archive, Report 2012/051. http://eprint.iacr.org/2012/051
9. Golić, J.D.: Cryptanalysis of alleged A5 stream cipher. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 239–255. Springer, Heidelberg (1997)
10. Shah, J., Mahalanobis, A.: A new guess-and-determine attack on the A5/1 stream cipher. Cryptology ePrint Archive, Report 2012/208. http://eprint.iacr.org/2012/208
11. Feng, X., Liu, J., Zhou, Z., Wu, Ch., Feng, D.: A byte-based guess and determine attack on sosemanuk. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 146–157. Springer, Heidelberg (2010)
12. Philip, H., Gregory, G.: Guess-and-determine attack on SNOW. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 37–46. Springer, Heidelberg (2003)
13. Sebastian, M., Alfredo, O.: Satellite Baseband firmware modifications. In: Ekoparty Security Conference, 2012. http://www.groundworkstech.com/blog/ekoparty2012satellitebasebandmods
14. Zeng, K., Yang, C.H., Rao, T.R.N.: On the linear consistency test (LCT) in cryptanalysis with applications. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 164–174. Springer, Heidelberg (1990)
15. Zhang, B., Feng, D.: New guess-and-determine attack on the self-shrinking generator. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 54–68. Springer, Heidelberg (2006)