Cryptanalysis of the DIZY Stream Cipher with Provable Security

Zhengting Li, Lin Ding, Xinhai Wang, Zheng Wu

***Abstract*—With the increasing deployment of resource-constrained devices in daily life, ultra-lightweight ciphers become a necessity to tackle the security and privacy concerns in resource-constrained devices. In 2023, Gül and Kara studied the question of how to design a secure ultra-lightweight stream cipher with a small internal state, and introduced a new small-state stream cipher called DIZY. The cipher utilizes Truncated Pseudorandom Permutations (TPP) and has a provable security in the indistinguishability model. It consists of two versions, called DIZY-128 with a 128-bit key and DIZY-80 with an 80-bit key, respectively. In this paper, effective key recovery attacks on DIZY-80 and DIZY-128 are proposed. Both attacks leverage the weakness of DIZY that the attacker can easily reach a weak state in the middle of the initialization using chosen IVs. Based on constructing Hellman tables, the key recovery attacks on DIZY-80 and DIZY-128 are further improved. The cryptanalytic results show that DIZY-80/DIZY-128 can only provide a 65/86-bit security level against the key recovery attack, while it is claimed to provide an 80/112-bit security level by the designers. Finally, an improved variant of DIZY, called DIZYa, is proposed. The analysis on DIZYa shows that the improved variant can provide better security resistance against all known attacks including our attacks on DIZY, while maintaining the commendable characteristics of DIZY. This makes DIZYa a more suitable small-state stream cipher choice for resource-constrained devices like RFID tags.**

***Index Terms*—Lightweight cryptography, cryptanalysis, DIZY, small-state stream cipher, Time-Memory-Data Tradeoff (TMDTO) attack.**

I. Introduction

THE deployment of resource-constrained devices such as sensor nodes, RFID tags, smart cards and industrial controllers is becoming more and more common. Since the conventional cryptographic algorithms were generally designed and optimized for desktop and server environments, it is difficult or even impossible to apply them in resource-constrained devices, due to their unacceptable implementation performance. To tackle new security and privacy concerns in resource-constrained devices, *lightweight cryptography* [1] emerges as the times require. Nowadays, it has become a subfield of cryptography that exclusively aims to provide solutions tailored for resource-constrained devices.

Manuscript received March 25, 2025; revised April 3, 2026; accepted May 9, 2026. This work was supported by the National Natural Science Foundation of China under Grant number 62472439. (*Corresponding author*: *Lin Ding*.)

Zhengting Li is with the Information Engineering University, Zhengzhou 450001, China (e-mail: lizhengting0225@163.com).

Lin Ding is with the Information Engineering University, Zhengzhou 450001, China (e-mail: dinglin_cipher@163.com).

Xinhai Wang is with the Information Engineering University, Zhengzhou 450001, China (e-mail: wxh1066559569@126.com).

Zheng Wu is with the Information Engineering University, Zhengzhou 450001, China (e-mail: g7001162@163.com).

Time-Memory-Data Tradeoff (TMDTO) attack [2], a generalization of Time-Memory Tradeoff (TMTO) attack [3], is a general cryptanalytic attack on symmetric ciphers. It gives a general principle in the design of stream ciphers [4], [5] that the internal state size is at least twice the key size. However, this is in stark contrast to the lightweight principle of stream ciphers, since a larger internal state necessarily increases resource requirements. At FSE 2015, Armknecht and Mikhalev [6] proposed a new framework to design stream ciphers with small internal state, called keystream generators with keyed update function, where the secret key is used in the internal state update to improve the resistance against TMDTO attacks. They used this new framework to design a stream cipher called Sprout, which is one of the first modern ultra-lightweight stream ciphers in academia with its hardware area cost less than 1000 GEs. After this, several ultra-lightweight stream ciphers have been proposed, e.g., Plantlet [7], Lizard [8], Atom [9] and Draco [10]. However, some of them have been proved to be insecure in [11]–[23]. Now, how to design a secure small-state stream cipher has become a research focus.

Recently, Gül and Kara [24] introduced a new construction method of diffusion layers for Substitution Permutation Network (SPN) structures along with its security proofs. They showed that these layers provide fast diffusion for the SPN ciphers having relatively large numbers of S-boxes (Substitution-boxes), in terms of the numbers of differentially and linearly active S-boxes. After that, they studied the question of how to design a secure small-state stream cipher without a keyed update function.As result, they introduced a new small-state stream cipher called DIZY through the utilization of Truncated Pseudorandom Permutations (TPP) [25], [26] and provided its security proofs against differential and linear cryptanalysis in the indistinguishability model. Two games were described by the designers to prove the indistinguishability of the TPP mode. According to the specification of DIZY, the cipher is provably secure against differential and linear cryptanalysis, and is claimed to secure against other known cryptanalytic attacks. The DIZY stream cipher consists of two versions, called DIZY-128 with a 128-bit key and DIZY-80 with an 80-bit key, respectively. In terms of area cost, power, and energy consumption, the hardware performance of DIZY is among the best in the existing small-state stream ciphers.

Related works. In the specification of DIZY [24], the designers claimed that DIZY-80 offers an 80-bit security level against the key recovery attack, and DIZY-128 offers a 112-bit security level against the key recovery attack in compliance with the NIST security requirements for the competition of

TABLE I
TMDTO ATTACKS ON DIZY

Stream Cipher	Online time complexity	Memory complexity	Data complexity	Offline time complexity	Ref.
DIZY-128	2^{85}	2^{85}	2^{32}	2^{128}	[24]
	$2^{65.60}$	$2^{102.58}$	2^{71}	$2^{97.32}$	This paper
	$2^{86.64}$	$2^{82.26}$	$2^{79.66}$	$2^{86.66}$	This paper
DIZY-80	2^{59}	2^{59}	2^{32}	2^{88}	[24]
	$2^{49.34}$	$2^{78.17}$	$2^{54.58}$	2^{73}	This paper
	2^{65}	$2^{63.17}$	$2^{61.02}$	2^{65}	This paper

lightweight algorithms. The designers showed that DIZY has a provable security in the indistinguishability model and enough security resistance against all known cryptanalytic attacks such as correlation attack, algebraic attack, Guess and Determine attack, differential attack, linear attack, integral attack, and boomerang attack. Since the maximum amount of keystream bits generated by one key-IV pair is limited to 2^{32} for DIZY, the designers showed that the offline time complexities of TMDTO attacks on DIZY-80 and DIZY-128 are at least 2^{88} and 2^{128}, respectively. Thus, the small-state stream cipher DIZY was claimed to be secure against TMDTO attacks by the designers. Up to now, no attack on DIZY has been published, except for the cryptanalytic results in the specification of DIZY.

Although the DIZY stream cipher had been published just a year before, the design of DIZY has attracted increasing attention of industry and academia. Supported by AMD under the Xilinx University Program, Schmid et al. [27] recently optimized the existing hardware implementation of DIZY and evaluated its energy efficiency. They claimed that the cipher is suitable for various applications, including Internet of Things (IoT) and embedded systems. From this we can see DIZY has already attracted the attention from some world-famous enterprises like AMD, due to its good hardware implementation performance. Hence, it is of great importance to analyze the security of DIZY deeply before putting it into practical applications.

Contributions. Since the security proof of DIZY is obtained by analyzing the numbers of differentially and linearly active S-boxes, the cipher is provably secure against differential and linear cryptanalysis, but not all cryptanalytic attacks. In the specification of DIZY, the security of DIZY against chosen IV attacks is not sufficiently considered by the designers. In this paper, an observation on the initialization of DIZY is presented, which reveals the fact that the attacker can easily reach a weak state in the middle of the initialization using chosen IVs. Based on this weakness, effective key recovery attacks on DIZY-80 and DIZY-128 are proposed. The complexities of these attacks are listed in Table I, and comparisons with the TMDTO attacks by the designers are made. As shown in Table I, the 80/128-bit key of DIZY-80/128 can be successfully recovered with complexities all below $2^{78.17}$/$2^{102.58}$, which are significantly better than the TMDTO attacks by the designers. Furthermore, we further improve the key recovery attacks on DIZY-80 and DIZY-128 based on constructing Hellman tables, which have complexities all below 2^{65} and $2^{86.66}$, respectively. The cryptanalytic results indicate that DIZY-DIZY-80/128 can only provide a 65/86-bit security level against the key recovery attack, while it is claimed to provide an 80/112-bit security level by the designers.

It is important to note that the keystream bits required by our attacks on DIZY-128 and DIZY-80 are generated by the fixed key together with different chosen IVs, which is completely reasonable in the chosen IV setting. For stream ciphers, the chosen IV setting is a standard and widely recognized attack setting, as almost all modern stream ciphers utilizes a public IV to realize the resynchronization mechanism. As pointed out in [28], [29], "*stream ciphers should be designed to resist attacks that exploit many known keystreams generated by the same key but different chosen IVs*." In the literature, a lot of attacks in the chosen IV setting on many well-known stream ciphers had been published, such as LEX [30], WG [30], WG-8 [31], Trivium [32], Grain v1 [33], Grain-128 [33], Grain-128a [34], ZUC [35], GEA-1/2 [36], ZUC-256 [37], DSC [38] and so on. Thus, the chosen IV setting we adopt in our attacks on DIZY-128 and DIZY-80 are completely realistic.

In the design specification of DIZY, the designers stipulates that the maximum amount of keystream bits generated by one key-IV pair is limited to 2^{32}. However, more than 2^{32} keystream bits generated by different key-IV pairs are acceptable. As for the data requirements of our attacks, the amounts of keystream bits required to be generated for one key-IV pair in our two attacks on DIZY-128 are 128 and $2^{15.66}$ respectively, and the amounts of keystream bits required to be generated for one key-IV pair in our two attacks on DIZY-80 are 96 and $2^{13.02}$ respectively. All of these are far less than the maximum amount (i.e., 2^{32}) stipulated by the designers. Thus, the data complexities of our attacks on DIZY-128 and DIZY-80 are completely valid. Although the total data complexities of our attacks are a little big for the ultra-lightweight DIZY stream cipher, the cryptanalytic results for the first time show that DIZY-80/DIZY-128 can not provide an 80/112-bit security level claimed by the designers.

Finally, an improved variant of DIZY, called DIZYa, is proposed. The analysis on DIZYa shows that the improved variant can provide better security resistance against all known cryptanalytic attacks including our attacks on DIZY, while maintaining the commendable characteristics of DIZY. This makes DIZYa a more suitable small-state stream cipher choice for resource-constrained devices like RFID tags.

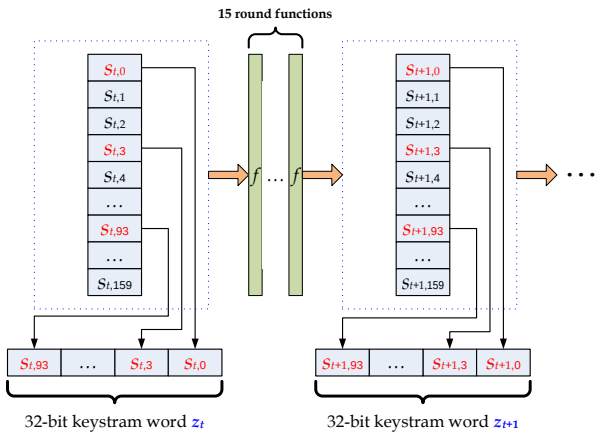

Fig. 1. The keystream generation of DIZY-128

Outline. In Section II, a brief description of DIZY is given. The key recovery attacks on DIZY-128 and DIZY-80 are introduced in Section III. The key recovery attacks on DIZY-128 and DIZY-80 are further improved in Section IV. In Section V, an improved variant of DIZY, called DIZYa, is proposed. Section VI concludes this paper.

II. A Brief Description of DIZY

The DIZY stream cipher consists of two versions, called DIZY-128 with a 128-bit key and DIZY-80 with an 80-bit key, respectively. We first introduce DIZY-128 and explain the differences of DIZY-80 from DIZY-128 later.

A. A Brief Description of DIZY-128

An overview of the keystream generation of DIZY-128 is depicted in Fig. 1. The internal state of DIZY-128 consists of 160 bits. Let $S^{(t)} = (s_{t,159}, \cdots, s_{t,0})$ be the 160-bit internal state of DIZY-128 at clock t. The state update function of DIZY-128 is an SPN structure which consists of 15 rounds, i.e.,

$$S^{(t+1)} = \overbrace{f \circ \cdots \circ f}^{15 \ rounds}\left(S^{(t)}\right)$$

where $S^{(t+1)}$ denotes the 160-bit internal state of DIZY-128 at clock $t+1$ and f denotes the round function used in the keystream generation of DIZY-128.

The round function of DIZY-128. Before introducing the round function f, we rewrite the 160-bit internal state $S_i^{(t)}$ after $i\,(0 \le i \le 15)$ iterations of the round function as $S_i^{(t)} = \left(IS_i^{(t)}[31], \cdots, IS_i^{(t)}[0]\right)$, where $IS_i^{(t)}[j] = \left(s_{i,5j+4}^{(t)}, s_{i,5j+3}^{(t)}, s_{i,5j+2}^{(t)}, s_{i,5j+1}^{(t)}, s_{i,5j}^{(t)}\right)(0 \le j \le 31)$ denotes a 5-bit word. The round function f consists of four operations, i.e., bit-wise XOR with a constant, an S-box layer, 4 binary matrix multiplications with an invertible binary matrix M and the permutation of eight sub-blocks.

More specifically, the round function f starts with the bit-wise XOR operation with a constant. The constant is generated by an LFSR of length 4, whose characteristic polynomial is $x^4 + x + 1$. In the i-th round, the LFSR generates a 4-bit word by clocking the LFSR i times starting from the initial value (1, 0, 0, 0) and then XORs the generated 4-bit word to the least significant four bits of each 5-bit word, i.e., for $j = 0, \cdots, 31$, $X_i^{(t)}[j] = \mathcal{XOR}_c\left(IS_i^{(t)}[j]\right) = \left(s_{i,5j+4}^{(t)}, \cdots, s_{i,5j+1}^{(t)}, s_{i,5j}^{(t)}\right) \oplus \left(0, LFSR^i(1,0,0,0)\right)$.

TABLE II
The 5-bit S-box used in the round function f

x	1f	1e	1d	1c	1b	1a	19	18
y	00	04	0e	09	0d	0b	1e	1b
x	17	16	15	14	13	12	11	10
y	1c	14	13	18	17	1d	05	0c
x	0f	0e	0d	0c	0b	0a	09	08
y	0f	11	08	15	03	1f	19	06
x	07	06	05	04	03	02	01	00
y	10	02	16	07	1a	0a	01	12

TABLE III
The permutation Γ used in the round function f

7	6	5	4	3	2	1	0
0	4	1	5	2	6	3	7

In the S-box layer, 32 5-bit S-boxes are performed on 32 words $X_i^{(t)} = \left(X_i^{(t)}[31], \cdots, X_i^{(t)}[0]\right)$ to generate a new 160-bit internal state $Y_i^{(t)} = \left(Y_i^{(t)}[31], \cdots, Y_i^{(t)}[0]\right)$, i.e., for $j = 0, \cdots, 31$,

$$Y_i^{(t)}[j] = Sbox\left(X_i^{(t)}[j]\right),$$

where the S-box takes a 5-bit word x as input and outputs a 5-bit word y, denoted as $y = Sbox(x)$. The 5-bit S-box is given in Table II.

In the next step, the 160-bit internal state is divided into 4 parts and each part consists of 40 bits. For each part, an invertible binary matrix M is multiplied to generate a new 160-bit internal state $W_i^{(t)}$, i.e.,

$$W_i^{(t)} = Mul_M\left(Y_i^{(t)}\right).$$

In the last step, the 160-bit internal state $W_i^{(t)}$ is divided into 8 sub-blocks, and then a permutation Γ, depicted in Table III, is applied to these 8 sub-blocks to generate the 160-bit internal state $S_{i+1}^{(t)}$, i.e.,

$$S_{i+1}^{(t)} = \Gamma\left(W_i^{(t)}\right).$$

The keystream output function of DIZY-128. At each clock, DIZY-128 selects 32 bits from the internal state $S_i^{(t)}$ as output, i.e.,

$$Z^{(t)} = (s_{t,93}, s_{t,90}, \cdots, s_{t,3}, s_{t,0}).$$

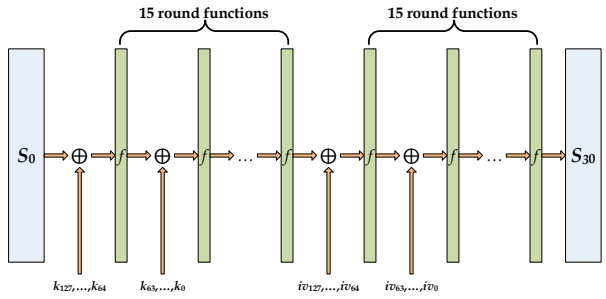

Fig. 2. The initialization of DIZY-128

where $Z^{(t)}$ denotes the 32-bit keystream word generated at clock t. It is produced by the least significant $3h$-th bits of the internal state $S_i^{(t)}$ for $h = 0, \cdots, 31$. It should be noted that the designers of DIZY stipulate that the maximum amount of keystream bits generated by one key-IV pair is limited to 2^{32} for DIZY-128 (and also DIZY-80).

The initialization of DIZY-128. DIZY-128 takes a 128-bit key and a 128-bit IV as input. The initialization of DIZY-128 consists of two phases, i.e., a key initialization phase and an IV initialization phase, see Fig. 2. Let $k = (k_{127}, \cdots, k_0)$ be the 128-bit secret key of DIZY-128. At first, the 160-bit initial state is set to be all zeros, i.e., $S_0^{(0)} = \left(s_{0,159}^{(0)}, \cdots, s_{0,0}^{(0)}\right) = (0, \cdots, 0)$. Then the most significant 64 bits of the 128-bit key (i.e., $k_{127}, \cdots, k_{64}$) are XORed with the 64-bit part of the initial state which is formed by concatenating the most significant two bits of each 5-bit word in order, i.e., for $j = 0, \cdots, 31$,

$$s_{0,5j+4}^{(0)} = k_{127-2\cdot(31-j)}$$
$$s_{0,5j+3}^{(0)} = k_{126-2\cdot(31-j)}$$

In the next step, the round function f is run once to update the 160-bit internal state $S_0^{(0)}$ to $S_1^{(0)}$ and the remaining key bits are incorporated similarly. The least significant 64 key bits (i.e., $k_{63}, \cdots, k_0$) are XORed two by two with the most significant two bits of the 5-bit words of $S_1^{(0)}$. Up to now, the 128-bit key has been completely incorporated and the obtained internal state is denoted as $S_{1'}^{(0)}$. The round function runs 14 times without incorporating any input afterward. Now the key initialization phase is completed at the end of the fifteenth round. After that, the IV initialization phase begins. In this phase, the 128-bit IV is incorporated with the same procedure as the key initialization phase, which requires 15 more rounds. That is, the initialization of DIZY-128 contains a total of 30 rounds. After these 30 initialization rounds, the keystream generation process of DIZY-128 begins.

B. A Brief Description of DIZY-80

The structure of DIZY-80 is completely the same of DIZY-128. Only the differences of DIZY-80 from DIZY-128 are briefly described here. DIZY-80 has a smaller internal state size of 120 bits. In the round function f, only 24 5-bit S-boxes are used in the S-box layer, and the invertible binary matrix M used in DIZY-80 is defined over a 30-bit part, rather than a 40-bit part in DIZY-128. DIZY-80 takes an 80-bit key and an 80-bit IV as input. Similar to the key initialization phase of DIZY-128, before the first round function, the most significant 48 bits of the key of DIZY-80 are XORed with the 48-bit part of the internal state, and then the remaining key bits are incorporated into the internal state after the first round function. The round function runs 14 times to complete the key initialization phase of DIZY-80 afterward. The IV is incorporated with the same procedure as the key initialization phase and the IV initialization phase is completed in 15 more rounds. Thus, the initialization of DIZY-80 also consists of 30 rounds.

III. A Key Recovery Attack on DIZY

In the specification of DIZY, the designers claimed that DIZY-128 offers a 112-bit security level against the key recovery attack in compliance with the NIST security requirements for the competition of lightweight algorithms. They utilized the TPP mode in the design of DIZY and provided its security proofs in the indistinguishability model. However, an observation on the initialization of DIZY will be presented in this section, which reveals the fact that the attacker can easily reach a weak state in the middle of the initialization using chosen IVs. Based on this weakness, effective key recovery attacks on DIZY-128 and DIZY-80 are proposed below. The cryptanalytic results show that DIZY-128 cannot offer a 112-bit security level against the key recovery attack, and DIZY-80 is not able to provide an 80-bit security level against the key recovery attack. We first introduce the key recovery attack on DIZY-128 and apply it to DIZY-80 later.

A. A Key Recovery Attack on DIZY-128

As shown in the description of DIZY-128 above, the initialization of DIZY-128 contains a total of 30 rounds. In the first 15 rounds, the 128-bit key is incorporated into the 160-bit internal state of DIZY-128 which is updated using the round function f. Let $S_{15}^{(0)} = \left(s_{15,159}^{(0)}, \cdots, s_{15,0}^{(0)}\right)$ be the 160-bit internal state of DIZY-128 after 15 rounds. It is easy to see that the 160-bit internal state $S_{15}^{(0)}$ is generated by taking the 128-bit key as input. In the remaining 15 rounds, the 128-bit IV is incorporated with the same procedure as the key initialization phase. Specifically, the most significant 64 bits of the 128-bit IV (i.e., $iv_{127}, \cdots, iv_{64}$) are XORed with the 64-bit part of $S_{15}^{(0)}$ which is formed by concatenating the most significant two bits of each 5-bit word in order, i.e., for $j = 0, \cdots, 31$,

$$s_{15',5j+4}^{(0)} = s_{15,5j+4}^{(0)} \oplus iv_{127-2\cdot(31-j)},$$
$$s_{15',5j+3}^{(0)} = s_{15,5j+3}^{(0)} \oplus iv_{126-2\cdot(31-j)}.$$

The generated 160-bit internal state is denoted as $S_{15'}^{(0)} = \left(s_{15',159}^{(0)}, \cdots, s_{15',0}^{(0)}\right)$, where the least significant three bits of each 5-bit word keep unchanged. In the next step, the round function f is run once to update the 160-bit internal state $S_{15'}^{(0)}$ to $S_{16}^{(0)}$ and the least significant 64 IV bits (i.e., $iv_{63}, \cdots, iv_0$) are XORed two by two with the most significant two bits of the 5-bit words of $S_{16}^{(0)}$. When taking a deep look at the

IV initialization phase of DIZY-128, an observation can be obtained as follows.

Observation 1. *For the 160-bit internal state* $S_{15'}^{(0)}$, $s_{15',5j+4}^{(0)} = 0$ *holds if* $s_{15,5j+4}^{(0)} = iv_{127-2\cdot(31-j)}$ *satisfies, and* $s_{15',5j+3}^{(0)} = 0$ *holds if* $s_{15,5j+3}^{(0)} = iv_{126-2\cdot(31-j)}$ *satisfies,* $j = 0, \cdots, 31$. *That is, for* $0 \le j \le 31$,

$$s_{15',5j+4}^{(0)} = 0 \Leftrightarrow s_{15,5j+4}^{(0)} = iv_{127-2\cdot(31-j)}$$
$$s_{15',5j+3}^{(0)} = 0 \Leftrightarrow s_{15,5j+3}^{(0)} = iv_{126-2\cdot(31-j)}$$

Generally, the IV is public and can be freely chosen by the attacker in the cryptanalysis of stream ciphers. By **Observation 1**, if the attacker exhausts the most significant $n\,(1 \le n \le 64)$ IV bits, i.e., $iv_{127}, \cdots, iv_{128-n}$, then there exists exactly one IV (out of all 2^n IVs) such that n bits of $S_{15'}^{(0)}$ are fixed to zeros. For example, when $n = 2$, the attacker exhausts the most significant 2 IV bits, i.e., iv_{127}, iv_{126}, then the two equations $s_{15,159}^{(0)} = iv_{127}$ and $s_{15,158}^{(0)} = iv_{126}$ will simultaneously hold for one out of $2^2 = 4$ IVs. For this exact IV, 2 bits of $S_{15'}^{(0)}$ will be fixed to zeros, i.e., $s_{15',159}^{(0)} = 0$ and $s_{15',158}^{(0)} = 0$ by **Observation 1**. We call this special internal state a *weak state*, whose formal definition is given as follows.

Definition 1. For a given positive integer n satisfying $1 \le n \le 64$, the internal state $S_{15'}^{(0)}$ is called a *weak state* if the following condition holds:

- If n is even, $s_{15',5j+4}^{(0)} = s_{15',5j+3}^{(0)} = 0$ holds for $32 - n/2 \le j \le 31$.
- If n is odd, $s_{15',5j+4}^{(0)} = 0$ holds for $32 - \lceil n/2 \rceil \le j \le 31$, and $s_{15',5j+3}^{(0)} = 0$ holds for $32 - \lfloor n/2 \rfloor \le j \le 31$.

For example, when $n = 31$, the internal state $S_{15'}^{(0)}$ is a *weak state* if $s_{15',5j+4}^{(0)} = 0$ holds for $16 \le j \le 31$ and $s_{15',5j+3}^{(0)} = 0$ holds for $17 \le j \le 31$. It is easy to see that a weak state is a special internal state $S_{15'}^{(0)}$, n bits out of which are fixed to zeros. Thus, the weak state belongs to a subset in size of 2^{160-n} internal states out of all 2^{160} internal states. By **Observation 1**, the attacker can obtain a weak state by trying 2^n chosen IVs. This weakness inspires us to recover a weak state by constructing a table, where some weak states and corresponding generated keystreams are stored. This is the basic idea of our key recovery attack on DIZY-128.

The attack on DIZY-128 consists of two processes, i.e., an offline process and an online process. In the offline process, the attacker constructs a table to store weak states, taking the generated keystream as an index. The procedure of the offline process is described as an algorithm, called **Algorithm 1**, as follows.

Algorithm 1 The procedure of the offline process

Output: A hash table Q

For $i = 1$ to m do the following:

- Select a weak state $\left\{\hat{S}_{15'}^{(0)}\right\}_i$, where n bits of $\left\{\hat{S}_{15'}^{(0)}\right\}_i$ are fixed to zeros and the remaining $160-n$ bits of $\left\{\hat{S}_{15'}^{(0)}\right\}_i$ are randomly chosen.
- Assume the weak state $\left\{\hat{S}_{15'}^{(0)}\right\}_i$ is the internal state $S_{15'}^{(0)}$ of DIZY-128, and then complete the IV initialization phase with $iv_{63} = \cdots = iv_0 = 0$.
- Generate $32l$-bit keystream $\hat{Z}_i = \left\{\hat{Z}^{(0)}, \cdots, \hat{Z}^{(l-1)}\right\}_i$ using $\left\{\hat{S}_{15'}^{(0)}\right\}_i$.
- Store the $160 - n$ bits of $\left\{\hat{S}_{15'}^{(0)}\right\}_i$ in a hash table Q indexed by $\hat{Z}_i$.

End for

Note that if $m > 2^{32l}$, there may be multiple weak states that produce the same $32l$-bit keystream. Hence the table cells should be equipped to accommodate multiple state vectors. The total memory space required to store the hash table Q is $M = m\,(160 - n)$ bits. For each of m randomly chosen weak states, **Algorithm 1** has to complete the IV initialization phase which consists of 15 rounds, and then generate $32l$-bit keystream $\hat{Z}_i$ which needs to run the round function $15l$ times. Thus, the total time cost required to generate the hash table Q can be calculated as $P = m \cdot (15 + 15l) = 15m\,(l + 1)$ DIZY rounds. The complexity unit 'DIZY round' refers to the time cost of executing the round function f once. To facilitate comparison, we use one encryption as a unit of time complexity throughout the paper. Since one encryption of DIZY contains at least 30 initialization rounds, thus the total time cost required to generate the hash table Q is at most

$$P = 15m\,(l + 1)/30 = m\,(l + 1)/2 \text{ encryptions.}$$

In the online process, the attacker utilizes d chosen IVs to ensure that at least one of all d key-IV pairs generates a weak state, and then recovers this weak state using the hash table Q built in the offline process. In this process, the attacker has to generate d different keystreams (i.e., $Z_1, \cdots, Z_d$) using the fixed unknown key together with d chosen IVs, i.e., $IV_1, \cdots, IV_d$. Here each of d keystreams consists of $32l$ bits, i.e., $|Z_j| = 32l$ for $1 \le j \le d$. For the attack to succeed, the parameter d should be an integral multiple of 2^n and the d IVs require to be carefully selected. Specifically, in each of the d IVs, the least significant 64 bits are fixed to zeros, i.e., $iv_i = 0$ for $0 \le i \le 63$. The most significant n bits $iv_{128-n}, \cdots, iv_{127}$ traverse all 2^n possible values, and the value of the remaining $64-n$ bits $iv_{64}, \cdots, iv_{127-n}$ has $d/2^n$ different options, each of which is randomly assigned. From this the attacker can obtain $2^n \cdot d/2^n = d$ chosen IVs. Thus, $2^n | d$ and $d \le 2^{64}$ simultaneously hold in this attack, and then about $d/2^n$ keystreams are possible to reach a collision in the hash table Q constructed in the offline process. By standard birthday assumptions, we will get a collision with a high probability when $m \cdot d/2^n \ge 2^{160-n}$. That is, $m \cdot d \ge 2^{160}$ should be satisfied. The procedure of the online process is described as an algorithm, called **Algorithm 2**, as follows.

Algorithm 2 The procedure of the online process

Input: The hash table Q and d different keystreams $Z_1, \cdots, Z_d$

Output: The $(160-n)$-bit vectors that can form weak states

For $j = 1$ to d do the following:

- Generate a keystream Z_j which has a size of $32l$ bits.
- Take the $32l$-bit keystream Z_j as an index and try to locate the corresponding entry in the hash table Q.
- If a match is found in the hash table Q, fetch and output the $160-n$ bits stored in the corresponding entry; otherwise, return to try the next $32l$-bit keystream.

End for

In the online process, d different keystreams have to be generated. Since the initialization of DIZY-128 contains 30 rounds totally and generating $32l$-bit keystream needs to run the round function $15l$ times, thus the time cost of generating d different keystreams is $d \cdot (30 + 15l) = 15d\,(l+2)$ DIZY rounds. After generating a keystream (say Z_j), a search in the hash table Q has to be executed. As a general rule, we assume that one memory access is comparable to one DIZY round. Thus, the total time complexity of the online process can be calculated as

$$\begin{aligned} T &= d \cdot (30 + 15l) + d = d\,(15l + 31) \text{ DIZY rounds} \\ &= d\,(15l + 31)/30 \text{ encryptions.} \end{aligned}$$

Once a weak state $S^{(0)}_{15'}$ is recovered by **Algorithm 2**, the attacker can execute the inverse round function 14 times to recover the internal state $S^{(0)}_{1'}$. In the forward direction, $S^{(0)}_{1'}$ is obtained from $S^{(0)}_0$ by executing the round function only once. The 128-bit key can be recovered from $S^{(0)}_{1'}$ by the following process.

Recall the initialization of DIZY-128, the 160-bit initial state of DIZY-128 is set to be all zeros, i.e., $S^{(0)}_0 = \left(s^{(0)}_{0,159}, \cdots, s^{(0)}_{0,0}\right) = (0, \cdots, 0)$. The attacker can guess the most significant 32 bits of the 128-bit key (i.e., $k_{127}, \cdots, k_{96}$), then the most significant 80 bits of $S^{(0)}_0$ become known to the attacker. It is easy to see that the significant 80 bits of the internal state $W^{(0)}_0$, which is obtained from $S^{(0)}_0$ after the first three operations of the round function f, is also known to the attacker. After applying the permutation Γ to the internal state $W^{(0)}_0$, the attacker can recover 4 sub-blocks (80 bits in total) of the internal state $S^{(0)}_1$. Using the known 4 sub-blocks (80 bits in total) and the internal state $S^{(0)}_{1'}$, the attacker can recover 32 key bits directly and check the correctness of the guessed value of $k_{127}, \cdots, k_{96}$. Since there are 48 bits that can be used for checking, the guessed 32 key bits $k_{127}, \cdots, k_{96}$ can be uniquely determined by simple comparisons. Thus, the attacker can recover 64 key bits of DIZY-128 by the key recovery process above. The time complexity of this process can be calculated as 2^{32} DIZY rounds, which is determined by the number of guessed key bits. Similarly, the attacker can recover the remaining 64 key bits by guessing the 32 key bits $k_{95}, \cdots, k_{64}$, which also leads to a time complexity of 2^{32} DIZY rounds. Thus, the 128-bit key can be recovered with a total time complexity of $2^{32} + 2^{32} = 2^{33}$ DIZY rounds, when the internal state $S^{(0)}_{1'}$ is obtained. Compared with the time cost of **Algorithm 2**, the time cost of recovering the 128-bit key from the obtained weak state is so small that can be neglected.

Up to now, we have introduced a key recovery attack on DIZY-128, which can successfully recover the 128-bit key of DIZY-128. The attack has an offline time complexity of $P = m\,(l+1)/2$ encryptions and an online time complexity of $T = d\,(15l+31)/30$ encryptions. It requires a memory space of $M = m\,(160-n)$ bits and has a data complexity of $D = 32ld$ bits. Note that $2^n | d$ and $d \leq 2^{64}$ should be simultaneously satisfied. To make the attack succeed with a high probability, $m \cdot d \geq 2^{160}$ should be satisfied, and the parameter l is set to be 4, which is large enough for the attack on DIZY-128. Let $n = 64, m = 2^{96}$ and $d = 2^{64}$, then a key recovery attack on DIZY-128 with $T = 2^{65.60}, M = 2^{102.58}, D = 2^{71}$ and $P = 2^{97.32}$ is obtained. It should be noted that only $32l = 128$ keystream bits are required to be generated for one key-IV pair in this attack, while the designers of DIZY stipulated that the maximum amount of keystream bits generated by one key-IV pair is limited to 2^{32}. Obviously, the data complexity of this attack on DIZY-128 is completely valid. This cryptanalytic result shows that the 128-bit key of DIZY-128 can be successfully recovered with complexities all below $2^{102.58}$ and thus DIZY-128 is not able to offer a 112-bit security level against the key recovery attack claimed by the designers.

B. A Key Recovery Attack on DIZY-80

Since DIZY-80 has a similar structure with DIZY-128, the attack on DIZY-128 above can be easily applied to DIZY-80. The key recovery attack on DIZY-80 has an offline time complexity of $P = m\,(l+1)/2$ encryptions, and an online time complexity of $T = d\,(15l+31)/30$ encryptions. Since DIZY-80 has an internal state size of 120 bits, the attack on the cipher requires a memory space of $M = m \cdot (120-n)$ bits and has a data complexity of $D = 32ld$ bits. Note that $2^n | d$ and $d \leq 2^{48}$ should be simultaneously satisfied, according to the key initialization phase of DIZY-80. Similarly, to make the attack on DIZY-80 succeed with a high probability, $m \cdot d \geq 2^{120}$ should be satisfied. Here, the parameter l is set to be 3, which is large enough for the attack on DIZY-80. Let $n = 48, m = 2^{72}$ and $d = 2^{48}$, then a key recovery attack on DIZY-80 with $T = 2^{49.34}, M = 2^{78.17}, D = 2^{54.58}$ and $P = 2^{73}$ is obtained. It should be noted that only $32l = 96$ keystream bits are required to be generated for one key-IV pair in this attack, which is far less than the maximum amount stipulated by the designers (i.e., 2^{32}). Obviously, the data complexity of this attack on DIZY-80 is completely valid. This cryptanalytic result shows that the 80-bit key of DIZY-80 can be successfully recovered with complexities all below $2^{78.17}$ and thus DIZY-80 is not able to offer an 80-bit security level against the key recovery attack claimed by the designers.

IV. Improved Key Recovery Attack on DIZY Based on Constructing Hellman Tables

In the section above, effective key recovery attacks on DIZY-128 and DIZY-80 have been presented. This section aims at proposing an improved key recovery attack on DIZY based on constructing Hellman tables. We first introduce the improved key recovery attack on DIZY-128 and apply it to DIZY-80 later.

A. Improved Key Recovery Attack on DIZY-128

Before introducing the improved attack on DIZY-128, a one-way function is constructed as follows.

$$\mathcal{F}:\{0,1\}^{160-n}\rightarrow\{0,1\}^{160-n}$$

The one-way function $\mathcal{F}$ is defined over $GF\left(2^{160-n}\right)$ and consists of the following operations.

- Firstly, the one-way function $\mathcal{F}$ takes a $(160-n)$-bit string (say $x\in\{0,1\}^{160-n}$) as input, and forms a weak state using x.
- Secondly, the one-way function $\mathcal{F}$ completes the IV initialization phase of DIZY-128 with $iv_{63}=\cdots=iv_0=0$.
- Thirdly, the one-way function $\mathcal{F}$ generates $32l$-bit keystream such that $32l\geq 160-n$.
- Finally, the one-way function $\mathcal{F}$ intercepts the front $160-n$ bits from the generated $32l$-bit keystream, and takes these $160-n$ bits as the output of $\mathcal{F}$.

Now, we aim at reversing the one-way function $\mathcal{F}$ constructed above to obtain a weak state. It is easy to see that the time complexity of executing the one-way function $\mathcal{F}$ once is about $15+15l$ DIZY rounds.

Let d be the number of chosen IVs used in the improved attack, and then the attacker can obtain a set of keystreams generated by the fixed key together with d chosen IVs. Specifically, in each of the d IVs, the least significant 64 bits are fixed to zeros, i.e., $iv_i=0$ for $0\leq i\leq 63$. The most significant n bits $iv_{128-n},\cdots,iv_{127}$ traverse all 2^n possible values, and the value of the remaining $64-n$ bits $iv_{64},\cdots,iv_{127-n}$ has $d/2^n$ different options, each of which is randomly assigned. From this the attacker can obtain $2^n\cdot d/2^n=d$ chosen IVs. Thus, $2^n|d$ and $d\leq 2^{64}$ simultaneously hold in the improved attack. Let d' be the length of the generated keystream by one key-IV pair. Since the designers of DIZY stipulated that the maximum amount of keystream bits generated by one key-IV pair is limited to 2^{32}, $32d'\leq 2^{32}$ should be satisfied and thus we have $d'\leq 2^{27}$. Denote by $\tilde{d}$ the maximal positive integer satisfying $\tilde{d}\leq d'$ and $32\left(d'-\tilde{d}\right)\geq 160-n$ simultaneously, then $\tilde{d}$ differential keystream segments, each of which is in size of $160-n$ bits, can be obtained in a keystream of length d'. Since the attack uses d keystreams of length d' generated by the fixed key together with d chosen IVs, thus a total of $\lambda=\left(d\cdot\tilde{d}\right)\Big/2^n$ keystream segments are possible to reach a weak state.

The improved key recovery attack on DIZY-128 also consists of two processes, i.e., an offline process and an online process. In the offline process, the attacker could construct t/λ tables, where $\lambda\leq t$ should be satisfied. Each table consists of m different starting points $x_1,\cdots,x_m$. For each of m different starting points, the attacker can apply the one-way function $\mathcal{F}$ to the starting point (say x_i) t times to form a chain as follows.

$$x_i\rightarrow\mathcal{F}\left(x_i\right)\rightarrow\mathcal{F}^2\left(x_i\right)\rightarrow\cdots\rightarrow y_i=\mathcal{F}^t\left(x_i\right)$$

The pairs (x_i,y_i) are stored in a hash table indexed by the value of y_i. To avoid collisions between the chains and the tables, t/λ variants of the one-way function $\mathcal{F}$, i.e., $\mathcal{F}_j\left(x\right)=L_j\left(\mathcal{F}\left(x\right)\right)$ $(1\leq j\leq t/\lambda)$, can be used. Here, $L_j\left(\cdot\right)$ is usually a simple output modification, e.g., reordering the bits of $\mathcal{F}\left(x\right)$. Thus the attacker can construct $(t/\lambda)\cdot m$ chains totally using t/λ functions $\mathcal{F}_1,\cdots,\mathcal{F}_{t/\lambda}$, m chains out of which are listed as follows.

$$x_1\rightarrow\mathcal{F}_j\left(x_1\right)\rightarrow\mathcal{F}_j^2\left(x_1\right)\rightarrow\cdots\rightarrow y_1=\mathcal{F}_j^t\left(x_1\right)$$

$$x_2\rightarrow\mathcal{F}_j\left(x_2\right)\rightarrow\mathcal{F}_j^2\left(x_2\right)\rightarrow\cdots\rightarrow y_2=\mathcal{F}_j^t\left(x_2\right)$$

$$\cdots\cdots$$

$$x_m\rightarrow\mathcal{F}_j\left(x_m\right)\rightarrow\mathcal{F}_j^2\left(x_m\right)\rightarrow\cdots\rightarrow y_m=\mathcal{F}_j^t\left(x_m\right)$$

By constructing the chains above, the offline process can cover a total of $(t/\lambda)\cdot t\cdot m=mt^2/\lambda$ points. Thus, the time complexity of the offline process can be calculated as

$$P=\left(mt^2/\lambda\right)\left(15+15l\right)=\left(15+15l\right)mt^2/\lambda \text{ DIZY rounds}$$
$$=\frac{(15+15l)mt^2/\lambda}{30}=\left(1+l\right)mt^2/2\lambda \text{ encryptions.}$$

In the offline process, the attacker has to construct t/λ tables (denoted as $Q_1,\cdots,Q_{t/\lambda}$) and each table consists of m pairs (x_i,y_i) $(1\leq i\leq m)$, which leads to a memory complexity of $M=2\left(160-n\right)mt/\lambda$ bits.

In the online process, since the attacker does not know which IV generates a weak state, the attacker has to try all $d\cdot\tilde{d}$ keystream segments to locate it. Among these keystream segments, about $\lambda=\left(d\cdot\tilde{d}\right)\Big/2^n$ keystream segments are possible to reach a weak state. For each keystream segment y, the attacker applies the one-way function $\mathcal{F}_j$ repeatedly to get the outputs $\mathcal{F}_j\left(y\right),\cdots,\mathcal{F}_j^t\left(y\right)$, and then checks whether the obtained output is an end point in the hash table Q_j. Once an output appears as an end point in the hash table Q_j, the attacker fetches the corresponding starting point and applies the one-way function $\mathcal{F}_j$ sequentially, until $y=\mathcal{F}_j\left(x\right)$ is reached. After that, the attacker can recover the 128-bit key using the obtained input x. Since the one-way function $\mathcal{F}_j$ should be executed t times for each keystream segment, the time complexity of the online process can be calculated as

$$T=(t/\lambda)\cdot t\cdot d\cdot\tilde{d}\cdot\left(15+15l\right) \text{ DIZY rounds}$$
$$=t^2\cdot 2^n\cdot\left(15+15l\right) \text{ DIZY rounds}$$
$$=\frac{t^2\cdot 2^n\cdot(15+15l)}{30} \text{ encryptions}$$
$$=t^2\cdot 2^n\cdot\left(1+l\right)/2 \text{ encryptions.}$$

Since the offline process has covered mt^2/λ points totally, the attacker can get a collision with a high probability by standard birthday assumptions when the following equality holds.

$$\left(mt^2/\lambda\right)\cdot\lambda = 2^{160-n}$$

That is, $mt^2 = 2^{160-n}$ should be satisfied. The data complexity of the improved key recovery attack on DIZY-128 is $D = d\cdot d'\cdot 32 = 32dd'$ bits. Similar to the key recovery attack on DIZY-128 in Section III, the parameter l is also set to be 4, which is large enough for the improved attack on DIZY-128. Let $n = 64, d = 2^{64}$ and $d' = 1621$, and then we obtain $\tilde{d} = 1618$ and $\lambda = 1618$. When we choose $m = 2^{74.68}$ and $t = 2^{10.66}$, an improved key recovery attack on DIZY-128 with $T = 2^{86.64}, M = 2^{82.26}, D = 2^{79.66}$ and $P = 2^{86.66}$ is obtained. It should be noted that only $32d' = 2^{15.66}$ keystream bits are required to be generated for one key-IV pair in this attack, while the designers of DIZY stipulated that the maximum amount of keystream bits generated by one key-IV pair is limited to 2^{32}. Obviously, the data complexity of the improved attack on DIZY-128 is completely valid.

Compared with the key recovery attack on DIZY-128 in Section III, the memory and offline time complexities are improved significantly, at the cost of increased online time and data complexities. More importantly, the improved key recovery attack on DIZY-128 based on constructing Hellman tables is $2^{102.58-86.66} = 2^{15.92}$ times better than the attack in Section III in terms of the complexity level. It also shows that DIZY-128 can only provide an 86-bit security level against the key recovery attack, rather than a 112-bit security level claimed by the designers.

B. Improved Key Recovery Attack on DIZY-80

Since DIZY-80 has a similar structure with DIZY-128, the improved key recovery attack on DIZY-128 can be easily applied to DIZY-80 by making some slight adjustments. Firstly, the attacker can construct a one-way function $\mathcal{G}: \{0,1\}^{120-n} \rightarrow \{0,1\}^{120-n}$ for DIZY-80, as DIZY-80 has an internal state size of 120 bits. Secondly, the attacker can implement similar offline and online processes as in the improved key recovery attack on DIZY-128. The attack has an online time complexity of $T = t^2\cdot 2^n\cdot(1+l)/2$ encryptions, an offline time complexity of $P = (1+l)\,mt^2/2\lambda$ encryptions and a data complexity of $D = 32dd'$ bits. The memory space required is about $M = 2\,(120-n)\,mt/\lambda$ bits, as DIZY-80 has an internal state size of 120 bits. Here, d' denotes the length of the generated keystream by one key-IV pair and $\tilde{d}$ denotes the maximal positive integer satisfying $\tilde{d} \le d'$ and $32\left(d'-\tilde{d}\right) \ge 120-n$ simultaneously. In the improved key recovery attack on DIZY-80, the restrictions $2^n | d$, $d \le 2^{48}$, $mt^2 = 2^{120-n}$ and $\lambda \le t$ should be satisfied simultaneously. The parameter l is set to be 3, which is large enough for the improved attack on DIZY-80. Let $n = 48, d = 2^{48}$ and $d' = 259$, and then we obtain $\tilde{d} = 256$ and $\lambda = 256$. When we choose $m = 2^{56}$ and $t = 2^8$, an improved key recovery attack on DIZY-80 with $T = 2^{65}, M = 2^{63.17}, D = 2^{61.02}$ and $P = 2^{65}$ is obtained. It should be noted that only $32d' = 2^{13.02}$ keystream bits are required to be generated for one key-IV pair in this attack, which is far less than the maximum amount stipulated by the designers (i.e., 2^{32}). Obviously, the data complexity of this attack on DIZY-80 is completely valid. The improved key recovery attack on DIZY-80 based on constructing Hellman tables is $2^{78.17-65} = 2^{13.17}$ times better than the attack in Section III in terms of the complexity level. It also shows that DIZY-80 can only provide a 65-bit security level against the key recovery attack, rather than an 80-bit security level claimed by the designers.

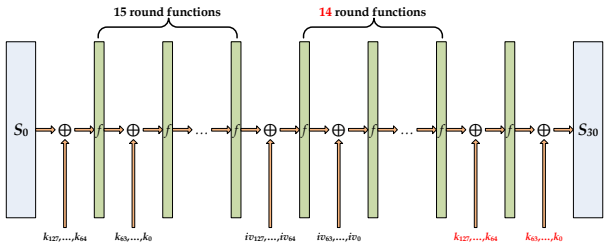

Fig. 3. The initialization of DIZY-128a

V. An Improved Variant of DIZY: DIZYa

Based on constructing Hellman tables, improved key recovery attacks on DIZY-128 and DIZY-80 have been proposed in the section above. The cryptanalytic results show that DIZY-128/DIZY-80 can only provide an 86/65-bit security level against the key recovery attack, while it is claimed to provide a 112/80-bit security level by the designers. Though the DIZY stream cipher is weak against our attacks, it still has several commendable characteristics, such as the introduction of TPP mode and the usage of an SPN structure to form the state update function. As analyzed by the designers, the DIZY stream cipher has a provable security in the indistinguishability model and enough security resistance against all known cryptanalytic attacks, e.g., correlation attack, algebraic attack, Guess and Determine attack, differential attack, linear attack, integral attack and boomerang attack. Besides, as for the hardware implementation performance, DIZY-128 has a hardware area cost of approximately 1.3K GE and is one of the most efficient lightweight ciphers. These commendable characteristics of DIZY motivate us to slightly modify it to get better security resistance against all known cryptanalytic attacks including our attacks, without losing its advantages.

A. The Design of DIZYa

Recall our attacks on DIZY above. The key making these attacks successful is that the attacker can reach a weak state in the middle of the initialization by using some chosen IVs. Thus, the least costly means to improve DIZY is to modify its initialization. To ensure that the attacker is not able to find the weak states in the middle of the initialization, XORing the 128-bit key into the internal state at the end of the initialization is a reasonable approach. The improved variant of DIZY, named DIZYa, has the same keystream generation as DIZY, while some slight modifications are made in the initialization. The DIZYa stream cipher also consists of two versions, called DIZY-128a with a 128-bit key and DIZY-80a with an 80-bit key, respectively.

Take DIZY-128a as an example. The initialization of DIZY-128a is depicted in Fig. 3, where the differences of DIZY-128a from DIZY-128 are clearly marked in red. The initialization of DIZY-128a also consists of 30 rounds, and the front 29 rounds are the same as in DIZY-128. Before the last round, the most significant 64 bits of the 128-bit key (i.e., $k_{127}, \cdots, k_{64}$) are XORed with the 64-bit part of the internal state which is formed by concatenating the most significant two bits of each 5-bit word in order. After that, the round function f is run once to update the 160-bit internal state and the remaining 64 key bits (i.e., $k_{63}, \cdots, k_0$) are incorporated similarly.

Similar to DIZY-128a, the improved version DIZY-80a can be easily obtained by modifying the initialization of DIZY-80 in the same manner.

B. The Advantages of DIZY-128a and DIZY-80a

Compared with the original versions, the improved versions DIZY-128a and DIZY-80a have several significant advantages, described as follows.

Firstly, DIZY-128a and DIZY-80a are secure against our attacks proposed above. Take DIZY-128a as an example. Though the attacker still can reach a weak state in the middle of the initialization, it becomes extremely difficult or even impossible to recover the weak state using the generated keystreams. That is because the 128-bit key is incorporated into the 160-bit internal state again in the last round. The attacker is not able to construct a one-way function from the weak states to the generated keystreams any more. Thus, the modifications make our attacks on DIZY invalid.

Secondly, DIZY-128a and DIZY-80a provide better security resistance against all known cryptanalytic attacks, such as TMDTO attack, correlation attack, algebraic attack, Guess and Determine attack, compared with the original versions. This claim can be demonstrated from the following three aspects.

- As analyzed in the specification of DIZY, the security proof of DIZY is obtained by analyzing the numbers of differentially and linearly active S-boxes. Since the round function is not changed in DIZYa, the provable security in the indistinguishability model keeps unchanged in DIZY-128a or DIZY-80a. Thus, DIZY-128a (or DIZY-80a) can provide the same provable security with DIZY-128 (or DIZY-80) against differential and linear cryptanalysis.
- In the specification of DIZY, the designers show that DIZY has enough security resistance against all known cryptanalytic attacks such as correlation attack, algebraic attack, Guess and Determine attack. Since the modifications are only made in the initialization process, the security resistance against these attacks that aim at recovering the internal state of the keystream generation process keeps unchanged in DIZY-128a or DIZY-80a. Thus, DIZY-128a (or DIZY-80a) can provide the same security resistance with DIZY-128 (or DIZY-80) against these known cryptanalytic attacks.
- For DIZY-128, DIZY-80 and many other stream ciphers (e.g., Trivium and Grain v1), the update function in the keystream generation or initialization process is invertible, and thus the attacker can compute the secret key easily from the recovered internal state. However, this is not applicable to DIZY-128a or DIZY-80a. Since the 128/80-bit key is incorporated into the 160/120-bit internal state of DIZY-128a/DIZY-80a again in the last round, the attacker is not able to compute the internal state in the front rounds backward from the recovered internal state in the keystream generation process. Therefore, for DIZY-128a or DIZY-80a an internal state recovery attack could not be converted to a key recovery attack, unless the 128 (or 80) key bits are guessed. Undoubtedly, this will enhance the security resistance against the cryptanalytic attacks that aim at recovering the internal state of the target cipher, such as TMDTO attack, correlation attack, algebraic attack, Guess and Determine attack.

Finally, the hardware implementation performance of DIZY-128a (or DIZY-80a) is almost the same of DIZY-128 (or DIZY-80). For the hardware encryption efficiency, DIZY-128a (or DIZY-80a) almost keeps the same as DIZY-128 (or DIZY-80), since the cost of the added 128-bit (or 80-bit) XOR operations with the key in the last initialization round are so small that can be negligible compared with the cost of 30 initialization rounds. For the hardware area cost, since the 128-bit (or 80-bit) XOR operations with the key have been implemented in the first initialization round of DIZY-128a (or DIZY-80a), the added 128-bit (or 80-bit) XOR operations with the key in the last initialization round can be easily implemented by reusing the existing gate circuits, and thus additional hardware area cost to implement DIZYa is able to be avoided. Therefore, DIZY-128a (or DIZY-80a) can still provide a good implementation performance in hardware without losing its provable security.

The analysis above shows that DIZY-128a and DIZY-80a can provide better security resistance against all known cryptanalytic attacks including our attacks on DIZY-128 and DIZY-80, while maintaining the commendable characteristics of DIZY. In view of the good performances of DIZY-128a and DIZY-80a in both security and hardware implementation, the improved versions are quite suitable in the resource-constrained devices, e.g., RFID tags, NFC, WiFi and Bluetooth.

VI. Conclusions

DIZY is a new small-state stream cipher designed by Gül and Kara in 2023. It utilizes Truncated Pseudorandom Permutations (TPP) and has a provable security in the indistinguishability model. Based on a discovered weakness of DIZY, effective key recovery attacks on DIZY-80 and DIZY-128 are proposed in this paper. The cryptanalytic results show that DIZY-80/DIZY-128 can only provide a 65/86-bit security level against the key recovery attack, while it is claimed to provide an 80/112-bit security level by the designers. Besides, an improved variant of DIZY, called DIZYa, is proposed. The analysis on DIZYa shows that the improved variant can provide better security resistance against all known attacks including our attacks on DIZY, while maintaining the commendable characteristics of DIZY. This makes DIZYa a more suitable small-state stream cipher choice for resource-constrained devices like RFID tags. We hope our cryptanalytic

results and new design can give some new insights into the design and analysis of secure small-state stream ciphers.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and the associate editor for their valuable comments and suggestions.

REFERENCES

[1] K. A. McKay, L. Bassham, M. S. Turan, N. Mouha, "Report on Lightweight Cryptography," in *National Institute of Standards and Technology Internal Report 8114*, March 2017 [Online]. https://doi.org/10.6028/NIST.IR.8114.

[2] A. Biryukov, A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," in *Proc. ASIACRYPT 2000*, Kyoto, Japan, Dec. 2000, pp. 1-13.

[3] M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, vol. 26, no. 4, pp. 401-406, July 1980.

[4] J. Hong, P. Sarkar, "New Applications of Time Memory Data Tradeoffs," in *Proc. ASIACRYPT 2005*, Chennai, India, Dec. 2005, pp. 353-372.

[5] O. Dunkelman, N. Keller, "Treatment of the initial value in Time-Memory-Data Tradeoff attacks on stream ciphers," *Information Processing Letters*, vol. 107, no. 5, pp. 133-137, Aug. 2008.

[6] F. Armknecht, V. Mikhalev, "On Lightweight Stream Ciphers with Shorter Internal States," in *Proc. Fast Software Encryption 2015*, Istanbul, Turkey, Mar. 2015, pp. 451-470.

[7] V. Mikhalev, F. Armknecht, C. Müller, "On Ciphers that Continuously Access the Non-Volatile Key," *IACR Transactions on Symmetric Cryptology*, vol. 2016, no. 2, pp. 52-79, 2016.

[8] M. Hamann, M. Krause, W. Meier, "LIZARD - A Lightweight Stream Cipher for Power-constrained Devices," *IACR Transactions on Symmetric Cryptology*, vol. 2017, no. 1, pp. 45-79, 2017.

[9] S. Banik, A. Caforio, T. Isobe, F. Liu, W. Meier, K. Sakamoto, S. Sarkar, "Atom: A stream cipher with double key filter," *IACR Transactions on Symmetric Cryptology*, vol. 2021, no. 1, pp. 5-36, 2021.

[10] M. Hamann, A. Moch, M. Krause, V. Mikhalev, "The Draco stream cipher: A power-efficient small-state stream cipher with full provable security against TMDTO attacks," *IACR Transactions on Symmetric Cryptology*, vol. 2022, no. 2, pp. 1-42, Jun. 2022.

[11] V. Lallemand, M. Naya-Plasencia, "Cryptanalysis of Full Sprout," in *Proc. CRYPTO 2015*, Santa Barbara, CA, USA, Aug. 2015, pp. 663-682.

[12] S. Banik, "Some Results on Sprout," in *Proc. INDOCRYPT 2015*, Bangalore, India, Dec. 2015, pp. 124-139.

[13] B. Zhang, X. Gong, "Another Tradeoff Attack on Sprout-Like Stream Ciphers," in *Proc. ASIACRYPT 2015*, Auckland, New Zealand, Nov. 2015, pp. 561-585.

[14] M. F. Esgin, O. Kara, "Practical Cryptanalysis of Full Sprout with TMD Tradeoff Attacks," in *Proc. Selected Areas in Cryptography - SAC 2015*, Sackville, NB, Canada, Aug. 2015, pp. 67-85.

[15] S. Banik, T. Isobe, T. Cui, J. Guo, "Some cryptanalytic results on Lizard," *IACR Transactions on Symmetric Cryptology*, vol. 2017, no. 4, pp. 82-98, 2017.

[16] B. Zhang, X. Gong, W. Meier, "Fast Correlation Attacks on Grain-like Small State Stream Ciphers," *IACR Transactions on Symmetric Cryptology*, vol. 2017, no. 4, pp. 58-81, 2017.

[17] S. Maitra, N. Sinha, A. Siddhanti, R. Anand, S. Gangopadhyay, "A TMDTO Attack Against Lizard," *IEEE Transactions on Computers*, vol. 67, no. 5, pp. 733-739, May 2018.

[18] M. Hamann, M. Krause, W. Meier, B. Zhang, "Design and analysis of small-state Grain-like stream ciphers," *Cryptography and Communications*, vol. 10, no. 5, pp. 803-834, 2018.

[19] Y. Todo, W. Meier, K. Aoki, "On the Data Limitation of Small-State Stream Ciphers: Correlation Attacks on Fruit-80 and Plantlet," in *Proc. Selected Areas in Cryptography - SAC 2019*, Waterloo, ON, Canada, Aug. 2019, pp. 365-392.

[20] S. Banik, K. Barooti, T. Isobe, "Cryptanalysis of Plantlet," *IACR Transactions on Symmetric Cryptology*, vol. 2019, no. 3, pp. 103-120, 2019.

[21] S. Banik, "Cryptanalysis of Draco," *IACR Transactions on Symmetric Cryptology*, vol. 2022, no. 4, pp. 92-104, 2022.

[22] O. Kara, M. F. Esgin, "On Analysis of Lightweight Stream Ciphers with Keyed Update," *IEEE Transactions on Computers*, vol. 68, no. 1, pp. 99-110, Jan. 2019.

[23] S. Kumar, S. Sarkar, "Conditional TMDTO as a MILP Instance," *IEEE Transactions on Information Theory*, vol. 69, no. 5, pp. 3330-3346, May 2023.

[24] Ç. Gül and O. Kara, "A New Construction Method for Keystream Generators," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3735-3744, Jun. 2023.

[25] W. Choi, B. Lee, J. Lee, "Indifferentiability of truncated random permutations," in *Proc. ASIACRYPT 2019*, Kobe, Japan, Dec. 2019, pp. 175-195.

[26] Y. Dodis, L. Reyzin, R. L. Rivest, E. Shen, "Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6," in *Proc. Fast Software Encryption 2009*, Leuven, Belgium, Feb. 2009, pp. 104-121.

[27] M. Schmid, T. Arul, E. B. Kavun, F. Regazzoni and O. Kara, "Robust and Energy-efficient Hardware Architectures for DIZY Stream Cipher," in *Proc. 2024 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Taipei, China, Nov. 2024, pp. 461-465.

[28] S. Fischer, K. Shahram, M. Willi, "Chosen IV statistical analysis for key recovery attacks on stream ciphers," in *Proc. AFRICACRYPT 2008*, Casablanca, Morocco, Jun. 2008, pp. 236-245.

[29] H. Englund, J. Thomas, S.T. Meltem, "A framework for chosen IV statistical analysis of stream ciphers," in *Proc. Indocrypt 2007*, Chennai, India, Dec. 2007, pp. 268-281.

[30] H. Wu, B. Preneel, "Resynchronization Attacks on WG and LEX," in *Proc. FSE 2006*, Graz, Austria, Mar. 2006, pp. 422-432.

[31] L. Ding, C. Jin, J. Guan, and Q. Wang, "Cryptanalysis of lightweight WG-8 stream cipher," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 645-652, Apr. 2014.

[32] I. Dinur, A. Shamir, "Cube attacks on tweakable black box polynomials," in *Proc. EUROCRYPT 2009*, Cologne, Germany, Apr. 2009, pp. 278-299.

[33] S. Knellwolf, M. Willi, N. María, "Conditional differential cryptanalysis of NLFSR-based cryptosystems," in *Proc. ASIACRYPT 2010*, Singapore, Dec. 2010, pp. 130-145.

[34] L. Ding and J. Guan, "Related key chosen IV attack on Grain-128a stream cipher," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 5, pp. 803-809, May. 2013.

[35] H. Wu, T. Huang, P.H. Nguyen, H. Wang, S. Ling, "Differential Attacks against Stream Cipher ZUC," in *Proc. ASIACRYPT 2012*, Beijing, China, Dec. 2012, pp. 262-277.

[36] L. Ding, Z. Wu, X. Wang, Z. Guan, M. Li, "New Attacks on the GPRS Encryption Algorithms GEA-1 and GEA-2," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2878-2889, Aug. 2022.

[37] F. Liu, M. Willi, S. Sarkar, G. Wang, R. Ito, T. Isobe, "New Cryptanalysis of ZUC-256 Initialization Using Modular Differences," *IACR Transactions on Symmetric Cryptology*, vol. 2022, no. 3, pp. 152-190, Sep. 2022.

[38] L. Ding, Z. Li, Z. Guan, X. Wang, Z. Wu, "Breaking the DECT Standard Cipher With Lower Time Cost," *IEEE Transactions on Computers*, vol. 73, no. 5, pp. 1290-1299, May. 2024.