Cryptanalysis of a Type of White-Box Implementations of the SM4 Block Cipher

Jiqiang Lu, Jingyu Li

***Abstract*—The SM4 block cipher was first released in 2006 as SMS4 used in the Chinese national standard WAPI, and became a Chinese national standard in 2016 and an ISO international standard in 2021. White-box cryptography aims primarily to protect the secret key used in a cryptographic software implementation in the white-box scenario that assumes an attacker to have full access to the execution environment and execution details of an implementation. Since white-box cryptography has many real-life applications nowadays, a few white-box implementations of the SM4 block cipher has been proposed with its increasingly wide use, among which a type of constructions is dominated, that use an affine diagonal block encoding to protect the original XOR sum of the three branches entering the S-box layer of a round and use its inverse to protect the original input of the S-box layer, such as Xiao and Lai's implementation in 2009, Shang's implementation in 2016 and Yao and Chen's implementation in 2020. In this paper, we show that this type of white-box SM4 constructions can be somewhat equivalent to a plain implementation mostly with Boolean masks from a security viewpoint, by devising collision-based attacks on Xiao and Lai's, Shang's and Yao and Chen's implementations with a time complexity of respectively about 2^{22}, 2^{39} and 2^{22} to peel off most white-box operations until only Boolean masks remain. Besides, we present a collision-based attack on a white-box SM4 implementation with a time complexity of about $2^{17.1}$ to recover an original round key, which uses a linear diagonal block encoding instead of an affine diagonal block encoding. Our results show that generating such a white-box SM4 implementation with affine encodings can be simplified into generating a plain implementation with Boolean masks (if its security expectation is beyond the above-mentioned complexity), and the effect of an affine encoding is significantly better than the effect of a linear encoding in the sense of our cryptanalysis results.**

***Index Terms*—White-box cryptography, SM4 (SMS4) block cipher, collision attack.**

I. Introduction

In 2002, Chow et al. [10], [11] introduced white-box cryptography and proposed white-box implementations of the AES [29] and DES [30] block ciphers. White-box cryptography works under the white-box security model, which assumes an attacker has full access to the execution environment and execution details (such as intermediate values, CPU calls, memory registers, etc) of a software implementation, giving the attacker more power than the black-box and grey-box security models. Nowadays, white-box cryptography has many real-life application scenarios like TV boxes, mobile phones and game consoles, and some white-box cryptography solutions have been in use.

The primary security threat for white-box cryptography is key extraction attack, which aims to extract the key used in white-box implementation. Chow et al.'s white-box AES implementation has been cryptanalysed extensively [6], [22], [28], [33], and the main attack results are as follows. In 2004, Billet et al. [6] presented an attack with a time complexity of 2^{30} (referred to below as BGE attack). In 2013, Lepoint et al. [22] improved the BGE attack to have a time complexity of 2^{22}, and presented a collision-based attack with a time complexity of 2^{22}. There are also a few attacks [18], [19], [25], [36] on Chow et al.'s white-box DES implementation. On the other hand, a number of different white-box implementation designs have been proposed [1], [3], [9], [20], [27], [37], but almost all of them have been broken with a practical or semi-practical time complexity [3], [12]–[14], [22]. Generally speaking, it has been well understood that the line of white-box implementation for an existing cryptographic algorithm is hardly possible to achieve the full security under the black-box model, but it is expected that it can still provide some protection with realistic significance.

The SM4 block cipher was first released in 2006 as the SMS4 [15] block cipher used in the Chinese national standard WAPI (WLAN Authentication and Privacy Infrastructure), which has a 128-bit block length and a 128-bit user key with a total of 32 rounds. SMS4 became a Chinese cryptographic industry standard in 2012, labeled with SM4, which then became a Chinese national standard [16] in 2016 and an ISO international standard in 2021 [17]. The main white-box implementation results of SMS4/SM4 are as follows. In 2009, Xiao and Lai [38] proposed the first white-box SM4 implementation in a relatively traditional way with a series of lookup tables and affine transformations. In 2013, Lin and Lai [23] attacked Xiao and Lai's white-box SM4 implementation with a time complexity of around 2^{47} by combining the BGE attack with other techniques like differential cryptanalysis [5]. In 2015, Shi et al. [32] proposed a lightweight white-box SM4 implementation based on the idea of dual cipher [4]. In 2016, Shang [31] improved Xiao and Lai's white-box SM4 implementation mainly by merging two individual lookup tables for two S-boxes into a larger whole, and got a security complexity of around 2^{48}; and Bai and Wu [2] proposed a white-box SM4 implementation with an S-

Manuscript received MONTH DAY, YEAR; revised MONTH DAY, YEAR. This work was supported by National Natural Science Foundation of China (No. 61972018) and Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS202102). (Corresponding author: Jiqiang Lu.)

Jiqiang Lu is with School of Cyber Science and Technology, Beihang University, Beijing 100191, China, with Guangxi Key Laboratory of Cryptography and Information Security, Guilin 541004, China, and also with Hangzhou Innovation Institute, Beihang University, Hangzhou 310053, China (e-mail: lvjiqiang@buaa.edu.cn)

Jingyu Li is with School of Cyber Science and Technology, Beihang University, Beijing 100191, China (e-mail: lijingyu98@buaa.edu.cn))

box input being divided into two shares. In 2018, Lin et al. [24] applied Biryukov et al.'s affine equivalence technique [7] to attack Shi et al.'s white-box SM4 implementation with a time complexity of 2^{49}. In 2020, Yao and Chen [39] proposed a white-box SM4 implementation with some original internal states expanded by dummy states under the control of a secret random number, and got the lowest attack complexity of about 2^{51} among a variety of attack techniques; and Wu et al. [35] proposed a white-box SM4 implementation with lookup tables and linear transformations, and showed it was resistant against BGE attack. In 2021, Wang et al. [34] applied Lepoint et al.'s collision-based idea to attack Shi et al.'s white-box SM4 implementation with a time complexity of around 2^{23} (note that this time complexity is obtained for checking the first-order derivatives for all 2^7 inputs under the worst case, and the time complexity under the expected case would be around $2^{17.1}$, as we do for Wu et al.'s white-box SM4 implementation in Section VI).

In this paper, we are concerned with Xiao and Lai's, Shang's, Yao and Chen's and Wu et al.'s white-box SM4 implementations, which are more or less different one another from a structural view but fundamentally all employ the construction method that uses an affine (or extremely even linear) diagonal block encoding to protect the original output of the XOR sum of the three branches entering the S-box layer of a round and uses the inverse of the encoding to protect the original input of the S-box layer. Especially, we focus on Yao and Chen's white-box SM4 implementation due to its representativeness, and apply Lepoint et al.'s collision-based idea to devise an attack with a total time complexity of about 2^{22} to peel off most white-box operations until only Boolean masks remain (but it seems impossible to work out the original key from the masked round keys), where the Boolean mask for a round key is associated with the affine diagonal block encoding; in particular, we first find that the effect of those dummy states can be bypassed without any workload by devising an appropriate collision function, then we find a trick to recover the linear parts of the affine diagonal block encodings at ease, next we recover the masked round key, and finally we recover both the linear (and constant sometimes) parts of the general affine encodings used. With more or less modifications due to their respective specifications, the attack is similarly applied to Xiao and Lai's and Shang's white-box SM4 implementations with a time complexity of about 2^{22} and 2^{39}, respectively, but can recover the original round key for Wu et al.'s white-box SM4 implementation with a time complexity of about $2^{17.1}$, since Wu et al.'s implementation uses a linear diagonal block encoding instead of an affine diagonal block encoding and so there is no mask at last. Our results show that generating such a white-box SM4 construction with affine encodings can be simplified by generating a plain implementation mostly with Boolean masks to avoid the cumbersome affine encodings, if its security expectation is beyond the above-mentioned complexity; and the effect of an affine encoding is significantly better than the effect of a linear encoding.

The remainder of the paper is organised as follows. We describe the notation and the SM4 block cipher in the next section, and present our cryptanalysis results on Yao and Chen's, Xiao and Lai's, Shang's and Wu et al.'s white-box SM4 implementations in Sections III to VI, respectively. Section VII concludes this paper.

II. Preliminaries

In this section, we give the notation used throughout this paper, and briefly describe the SM4 block cipher.

A. Notation

We use the following notation throughout this paper.

$\oplus$ bitwise exclusive OR (XOR)
$\gg$ right shift of a bit string
$\lll$ left rotation of a bit string
$||$ bit string concatenation
$\circ$ functional composition

B. The SM4 Block Cipher

SM4 [15], [16] is a generalised Feistel cipher with 32 rounds, a 128-bit block size and a 128-bit key length. Denote by $(X_i, X_{i+1}, X_{i+2}, X_{i+3})$ the 128-bit input to the i-th round, by rk_i the 32-bit i-th round key, where $X_i \in \mathrm{GF}(2)^{32}$ and $i = 0, 1, \ldots, 31$.

Define the nonlinear function $\tau : \mathrm{GF}(2)^{32} \to \mathrm{GF}(2)^{32}$ that applies the same 8-bit S-box $\mathbf{S}$ four times in parallel as

$$x \mapsto \big(\mathbf{S}(x_{[31\ldots24]}), \mathbf{S}(x_{[23\ldots16]}), \mathbf{S}(x_{[15\ldots8]}), \mathbf{S}(x_{[7\ldots0]})\big);$$

and define the linear function $\mathbf{L} : \mathrm{GF}(2)^{32} \to \mathrm{GF}(2)^{32}$ as

$$x \mapsto x \oplus (x \lll 2) \oplus (x \lll 10) \oplus (x \lll 18) \oplus (x \lll 24). \quad (1)$$

Then, the invertible transformation $\mathbf{T} : \mathrm{GF}(2)^{32} \times \mathrm{GF}(2)^{32} \to \mathrm{GF}(2)^{32}$ is defined to be

$$(x, rk_i) \to \mathbf{L}(\tau(x \oplus rk_i)),$$

and the round function $\mathbf{F} : \mathrm{GF}(2)^{128} \times \mathrm{GF}(2)^{32} \to \mathrm{GF}(2)^{128}$ under round key rk_i is

$$\begin{aligned}&((X_i, X_{i+1}, X_{i+2}, X_{i+3}), rk_i) \mapsto (X_{i+1}, X_{i+2}, X_{i+3},\\ &X_i \oplus \mathbf{T}\big(X_{i+1} \oplus X_{i+2} \oplus X_{i+3}, rk_i\big)\big). \quad (2)\end{aligned}$$

The encryption procedure of SM4, as depicted in Fig. 1, consists of the 32 round functions $\mathbf{F}$'s and finally a reverse transformation $R : \mathrm{GF}(2)^{128} \to \mathrm{GF}(2)^{128}$ defined as

$$(X_{32}, X_{33}, X_{34}, X_{35}) \mapsto (X_{35}, X_{34}, X_{33}, X_{32}).$$

The decryption process of SM4 is the same as the encryption process, except that the round keys are used in the reverse order. We refer the reader to [15], [16] for detailed specifications.

Particularly, it is easy and worthy to note that the linear transformation $\mathbf{L}$ (as described in Eq. (1)) of SM4 can also be represented as an invertible 32×32-bit matrix

$$\begin{bmatrix} B_1 & B_2 & B_2 & B_3 \\ B_3 & B_1 & B_2 & B_2 \\ B_2 & B_3 & B_1 & B_2 \\ B_2 & B_2 & B_3 & B_1 \end{bmatrix}, \quad (3)$$

with B_1, B_2 and B_3 being invertible 8×8-bit block matrices.

Let x_0, x_1, x_2, x_3 be four byte variables, represent $\mathbf{L}$ as four 32×8-bit matrices $\begin{bmatrix}\mathbf{L}_0 & \mathbf{L}_1 & \mathbf{L}_2 & \mathbf{L}_3\end{bmatrix}$, and define

$$\begin{aligned}
\mathbf{L}_0(x) &= x \cdot \begin{bmatrix}B_1 & B_3 & B_2 & B_2\end{bmatrix}^T, \\
\mathbf{L}_1(x) &= x \cdot \begin{bmatrix}B_2 & B_1 & B_3 & B_2\end{bmatrix}^T, \\
\mathbf{L}_2(x) &= x \cdot \begin{bmatrix}B_2 & B_2 & B_1 & B_3\end{bmatrix}^T, \\
\mathbf{L}_3(x) &= x \cdot \begin{bmatrix}B_3 & B_2 & B_2 & B_1\end{bmatrix}^T,
\end{aligned}$$

then $\mathbf{L}(x_0||x_1||x_2||x_3) = \mathbf{L}_0(x_0) \oplus \mathbf{L}_1(x_1) \oplus \mathbf{L}_2(x_2) \oplus \mathbf{L}_3(x_3)$.

III. Cryptanalysis of Yao and Chen's White-Box SM4 Implementation

In this section, we first describe Yao and Chen's white-box SM4 implementation, and then present our cryptanalysis result on it.

A. Yao and Chen's White-Box SM4 Implementation

Yao and Chen's white-box SM4 implementation [39] is based on internal state expansion, particularly, the 32×32-bit matrix representation described in Eq. (3) of the linear transformation $\mathbf{L}$ is expanded to the following 64×64-bit matrix $\widehat{\mathbf{L}}$ with the 8×8-bit zero matrix $\mathbf{0}$:

$$\widehat{\mathbf{L}} = \begin{bmatrix}
B_1 & \mathbf{0} & B_2 & \mathbf{0} & B_2 & \mathbf{0} & B_3 & \mathbf{0} \\
\mathbf{0} & B_1 & \mathbf{0} & B_2 & \mathbf{0} & B_2 & \mathbf{0} & B_3 \\
B_3 & \mathbf{0} & B_1 & \mathbf{0} & B_2 & \mathbf{0} & B_2 & \mathbf{0} \\
\mathbf{0} & B_3 & \mathbf{0} & B_1 & \mathbf{0} & B_2 & \mathbf{0} & B_2 \\
B_2 & \mathbf{0} & B_3 & \mathbf{0} & B_1 & \mathbf{0} & B_2 & \mathbf{0} \\
\mathbf{0} & B_2 & \mathbf{0} & B_3 & \mathbf{0} & B_1 & \mathbf{0} & B_2 \\
B_2 & \mathbf{0} & B_2 & \mathbf{0} & B_3 & \mathbf{0} & B_1 & \mathbf{0} \\
\mathbf{0} & B_2 & \mathbf{0} & B_2 & \mathbf{0} & B_3 & \mathbf{0} & B_1
\end{bmatrix}.$$

Represent the matrix $\widehat{\mathbf{L}}$ as four 64×16-bit matrices, that is, $\widehat{\mathbf{L}} = \begin{bmatrix}\widehat{\mathbf{L}}_0 & \widehat{\mathbf{L}}_1 & \widehat{\mathbf{L}}_2 & \widehat{\mathbf{L}}_3\end{bmatrix}$. Then, an encryption round of Yao and Chen's white-box SM4 implementation consists of the following three parts according to Eq. (2), as depicted in Fig. 2. Note first that X_l is the corresponding original value protected with an affine output encoding $P_l(x) = A_l \cdot x \oplus a_l$, where x is a 32-bit variable, the linear part A_l is a secret (randomly generated) general invertible 32×32-bit matrix, the constant part a_l is a secret (randomly generated) 32-bit vector, and $l = 0, 1, \cdots, 35$.

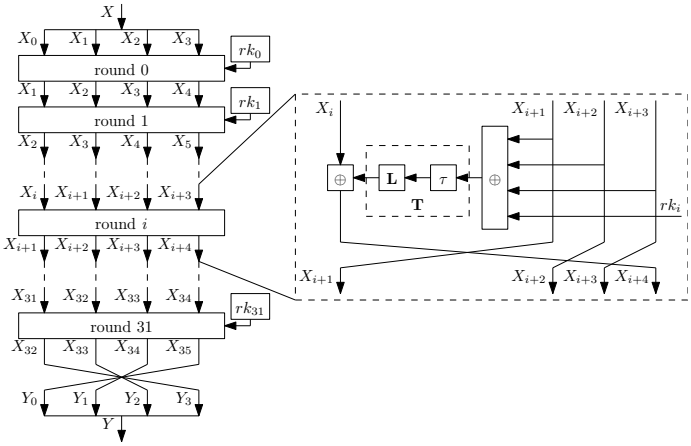

Figure 1. SM4 encryption procedure

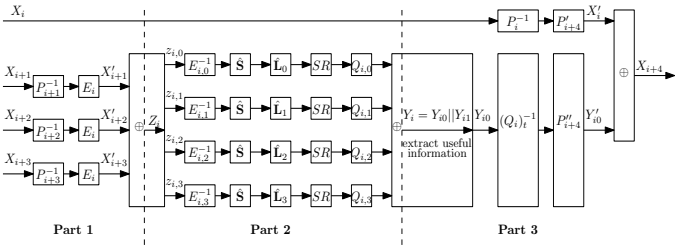

Figure 2. An encryption round of Yao and Chen's white-box SM4 implementation

1) Part 1 – Implement $X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \mapsto Z_i$: In order to obtain the original value of $X_{i+1} \oplus X_{i+2} \oplus X_{i+3}$ from the protected forms X_{i+1}, X_{i+2} and X_{i+3}, apply first the inverses P_{i+1}^{-1}, P_{i+2}^{-1} and P_{i+3}^{-1} of the three output encodings respectively to X_{i+1}, X_{i+2} and X_{i+3}, followed by an identical diagonal output encoding $E_i = \mathrm{diag}(E_{i,0}, E_{i,1}, E_{i,2}, E_{i,3})$, where $E_{i,0}, E_{i,1}, E_{i,2}, E_{i,3}$ are four general invertible 8×8-bit affine transformations ($i = 0, 1, \ldots, 31$).

This part can be summarised as

$$\begin{aligned}
X'_{i+j} &= E_i \circ P_{i+j}^{-1}(X_{i+j}), \quad j = 1, 2, 3; \\
Z_i &= X'_{i+1} \oplus X'_{i+2} \oplus X'_{i+3},
\end{aligned}$$

where Z_i is a 32-bit variable. Observe that the final result of this part $Z_i = E_i \circ (P_{i+1}^{-1}(X_{i+1}) \oplus P_{i+2}^{-1}(X_{i+2}) \oplus P_{i+3}^{-1}(X_{i+3}))$ is the original value of $X_{i+1} \oplus X_{i+2} \oplus X_{i+3}$ protected with the output encoding E_i in such a way that its four bytes are protected respectively with the four 8-bit encodings $E_{i,0}, E_{i,1}, E_{i,2}$ and $E_{i,3}$.

2) Part 2 – Implement $\mathbf{T}(Z_i, rk_i) \mapsto Y_i (= Y_{i0}||Y_{i1})$: The input Z_i of the second part is the output of the first part, represent Z_i as 4 bytes $Z_i = (z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3})$, and represent the round key rk_i as 4 bytes $rk_i = (rk_{i,0}, rk_{i,1}, rk_{i,2}, rk_{i,3})$, where $i = 0, 1, \ldots, 31$. Next, construct four lookup tables that map from 8-bit input to 64-bit output each, as follow:

$$\begin{aligned}
Table_{i,0} &= G_{i,0} \circ \widehat{\mathbf{L}}_0[\widehat{\mathbf{S}}(E_{i,0}^{-1}(z_{i,0}), rk_{i,0}, \alpha_{i,0})_{t_{i,0}}], \\
Table_{i,1} &= G_{i,1} \circ \widehat{\mathbf{L}}_1[\widehat{\mathbf{S}}(E_{i,1}^{-1}(z_{i,1}), rk_{i,1}, \alpha_{i,1})_{t_{i,1}}], \\
Table_{i,2} &= G_{i,2} \circ \widehat{\mathbf{L}}_2[\widehat{\mathbf{S}}(E_{i,2}^{-1}(z_{i,2}), rk_{i,2}, \alpha_{i,2})_{t_{i,2}}], \\
Table_{i,3} &= G_{i,3} \circ \widehat{\mathbf{L}}_3[\widehat{\mathbf{S}}(E_{i,3}^{-1}(z_{i,3}), rk_{i,3}, \alpha_{i,3})_{t_{i,3}}],
\end{aligned}$$

where

- $\alpha_{i,j}$ is an 8-bit random number ($j = 0, 1, 2, 3$);
- $\widehat{\mathbf{L}}_j$ is the corresponding j-th 64×16-bit part of $\widehat{\mathbf{L}}$;
- $(t_{i,0}, t_{i,1}, t_{i,2}, t_{i,3})$ $(t_{i,j} \in \{0, 1\})$ is a 4-bit random vector, and

$$\widehat{\mathbf{S}}(E_{i,j}^{-1}(z_{i,j}), rk_{i,j}, \alpha_{i,j})_{t_{i,j}} = \begin{cases}
\mathbf{S}(E_{i,j}^{-1}(z_{i,j}) \oplus rk_{i,j}) \;||\; \mathbf{S}(E_{i,j}^{-1}(z_{i,j}) \oplus \alpha_{i,j}), & t_{i,j} = 0; \\
\mathbf{S}(E_{i,j}^{-1}(z_{i,j}) \oplus \alpha_{i,j}) \;||\; \mathbf{S}(E_{i,j}^{-1}(z_{i,j}) \oplus rk_{i,j}), & t_{i,j} = 1.
\end{cases}$$

That is, the $\widehat{\mathbf{S}}$ operation is constructed by expanding the original $\mathbf{S}$ operation with a dummy $\mathbf{S}$ operation under the control of the 1-bit $t_{i,j}$ parameter.

- $G_{i,j}$ is the composition of a shift matrix SR and an output encoding $Q_{i,j}$. The shift matrix SR transforms the expanded 64-bit value after $\widehat{\mathbf{L}}_j$ into such a 64-bit value that the former half is the original 32-bit part (without expansion) and the latter half consists only of some dummy bits. $Q_{i,j}$ is of the affine form $Q_{i,j}(x) = L_Q \cdot x \oplus C_{Q_{i,j}}$, here x is a 64-bit variable, the linear part L_Q is a block diagonal matrix being composed of eight 8×8-bit matrices, and the constant part $C_{Q_{i,j}}$ consists of eight concatenated 8-bit vectors.

The final output of this part is the XOR of the four 64-bit outputs of the four lookup tables, which is denoted by $Y_i = Y_{i0}||Y_{i1}$ with Y_{i0} being supposed to be the original useful 32-bit value.

3) Part 3 – Implement $Y_{i0} \oplus X_i \mapsto X_{i+4}$: This part first extracts the original useful 32-bit value from the 64-bit expanded output of the second part, and then calculates X_{i+4}, as follows.

$$\begin{aligned} Y'_{i0} &= P''_{i+4} \circ (Q_i)_t^{-1}(Y_{i0}), \\ X'_i &= P'_{i+4} \circ P_i^{-1}(X_i), \\ X_{i+4} &= Y'_{i0} \oplus X'_i, \end{aligned}$$

where $(Q_i)_t^{-1}$ represents the corresponding part of the inverse of the encodings $L_Q \cdot x \oplus (C_{Q_{i,0}} \oplus C_{Q_{i,1}} \oplus C_{Q_{i,2}} \oplus C_{Q_{i,3}})$ of the second part, and P'_{i+4} and P''_{i+4} are new affine output encodings of the forms $P'_{i+4}(x) = P_{i+4} \oplus a'_{i+4}$ and $P''_{i+4}(x) = P_{i+4} \oplus a''_{i+4}$, respectively, so that X_{i+4} is a protected form with an affine output encoding $P_{i+4}(x) = A_{i+4} \cdot x \oplus a_{i+4}$, like X_i.

As a result, the whole white-box SM4 implementation can be obtained by iterating the above process for all the 32 rounds with possibly independent encodings.

Yao and Chen analysed its security against a variety of attack techniques like BGE, and got that the attack complexity using affine equivalence technique was 2^{97}, and the lowest attack complexity was 2^{51} among all used attack techniques.

B. Cryptanalysis of Yao and Chen's Implementation

In this subsection, we apply Lepoint et al.'s collision-based attack idea to peel off most white-box operations until only Boolean masks remain for Yao and Chen's white-box SM4 implementation, with a time complexity of about $2^{24.4}$. AES and SM4 have different structures, and Yao and Chen's white-box SM4 implementation is distinct from Chow et al.'s white-box AES implementation: there are dummy states with indeterminate positions and the encoding used in X_{i+4} involves a general 32×32-bit matrix, which does not allow us to apply Lepoint et al.'s attack idea efficiently within one round, as for Chow et al.'s white-box AES implementation. However, after a detailed investigation we find an appropriate collision function by considering two consecutive rounds in Yao and Chen's white-box SM4 implementation, plus a trick that can recover the linear parts of the concerned encodings, to bypass the effects due to the dummy states and etc.

1) Devising a Collision Function: As illustrated in Fig. 3 at a high level, the collision function used in our attack takes as input the two 32-bit input parameters $(z_{i,0}||z_{i,1}||z_{i,2}||z_{i,3}, X_i)$

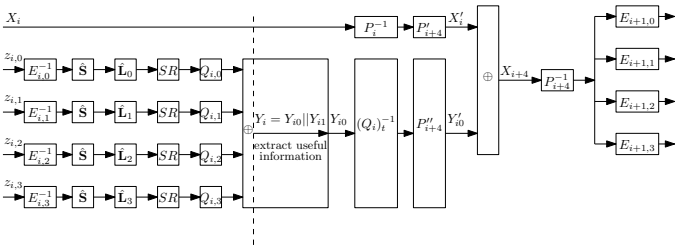

Figure 3. Our collision function on Yao and Chen's white-box SM4 implementation

in the second part of an encryption round of Yao and Chen's white-box SM4 implementation, and ends with the output of an $E_{i+1,j}$ operation of the X_{i+4} branch in the first part of the next encryption round ($j = 0, 1, 2, 3$). Observe that E_i and E_{i+1} are diagonal affine transformations, $E_{i,j}$ and $E_{i+1,j}$ are invertible 8×8-bit affine transformations, and $z_{i,j}$ is the original input byte to the j-th original S-box of the i-th encryption round in a protected form with $E_{i,j}$.

The collision function is functionally equivalent and can be simplified to the one depicted in Fig. 4. In our attack and all subsequent descriptions, we set X_i such that $P'_{i+4} \circ P_i^{-1}(X_i) = 0$, and denote the constant $A_{i+4}^{-1} \cdot a''_{i+4} \oplus A_{i+4}^{-1} \circ P'_{i+4} \circ P_i^{-1}(X_i) = A_{i+4}^{-1} \cdot a''_{i+4}$ by ε_i. We now explain where the value ε_i comes from. Let $\widehat{Y}_i$ denotes the original 32-bit value immediately after the $\mathbf{L}$ operation under the input $Z_i = (z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3})$, then we have

$$\begin{aligned} & P_{i+4}^{-1} \circ (Y'_{i0} \oplus P'_{i+4} \circ P_i^{-1}(X_i)) \\ &= P_{i+4}^{-1}(Y'_{i0}) \oplus A_{i+4} \circ P'_{i+4} \circ P_i^{-1}(X_i) \\ &= P_{i+4}^{-1} \circ P''_{i+4}(\widehat{Y}_i) \\ &= P_{i+4}^{-1} \circ (P_{i+4}(\widehat{Y}_i) \oplus a''_{i+4}) \\ &= P_{i+4}^{-1} \circ (A_{i+4}(\widehat{Y}_i) \oplus a_{i+4} \oplus a''_{i+4}) \\ &= A_{i+4}^{-1} \circ (A_{i+4}(\widehat{Y}_i) \oplus a_{i+4} \oplus a''_{i+4} \oplus a_{i+4}) \\ &= \widehat{Y}_i \oplus A_{i+4}^{-1} \cdot a''_{i+4}, \end{aligned}$$

which is equal to $\widehat{Y}_i \oplus \varepsilon_i$ under $P'_{i+4} \circ P_i^{-1}(X_i) = 0$.

As a consequence, the collision function denoted by $f^i(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}, X_i)$, or simply $f^i(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3})$ under $P'_{i+4} \circ P_i^{-1}(X_i) = 0$, is

$$\begin{aligned} & f^i(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}) \\ &= \begin{bmatrix} E_{i+1,0} \\ E_{i+1,1} \\ E_{i+1,2} \\ E_{i+1,3} \end{bmatrix} \circ \oplus_{\varepsilon_i} \circ \mathbf{L} \circ \begin{bmatrix} \mathbf{S} \circ \oplus_{rk_{i,0}} \circ E_{i,0}^{-1}(z_{i,0}) \\ \mathbf{S} \circ \oplus_{rk_{i,1}} \circ E_{i,1}^{-1}(z_{i,1}) \\ \mathbf{S} \circ \oplus_{rk_{i,2}} \circ E_{i,2}^{-1}(z_{i,2}) \\ \mathbf{S} \circ \oplus_{rk_{i,3}} \circ E_{i,3}^{-1}(z_{i,3}) \end{bmatrix}. \end{aligned}$$

Furthermore, we express f^i as a concatenation of four byte functions f_0^i, f_1^i, f_2^i and f_3^i:

$$\begin{aligned} & f^i(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}) \\ =& [f_0^i(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}), f_1^i(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}), \\ & f_2^i(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}), f_3^i(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3})]^T; \end{aligned}$$

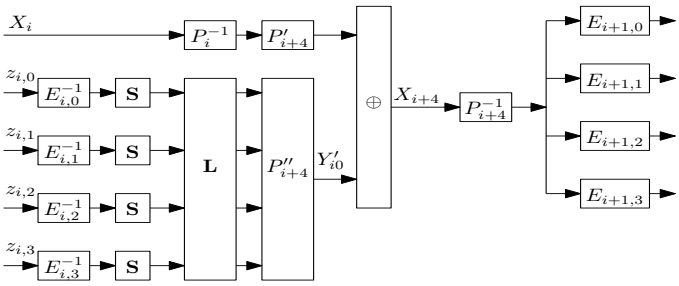

Figure 4. Equivalent of collision function on Yao and Chen's white-box SM4 implementation

and define $\mathbf{S}_j$ function as

$$\begin{aligned}\mathbf{S}_j^i(\cdot) &= \mathbf{S} \circ \oplus_{rk_{i,j}} \circ E_{i,j}^{-1}(\cdot) \\ &= \mathbf{S}(rk_{i,j} \oplus E_{i,j}^{-1}(\cdot)), \quad j = 0,1,2,3. \end{aligned} \tag{4}$$

2) Recovering $\mathbf{S}_j^i$ Functions: Next we try to recover the functions $\mathbf{S}_0^i$, $\mathbf{S}_1^i$, $\mathbf{S}_2^i$ and $\mathbf{S}_3^i$ by exploiting collisions on the output of the functions f_j^i. We first use the following collision to recover $\mathbf{S}_0^i$ and $\mathbf{S}_1^i$:

$$f_0^i(\alpha,0,0,0) = f_0^i(0,\beta,0,0), \tag{5}$$

where $\alpha,\beta \in \mathrm{GF}(2)^8$. By the linear transformation $\mathbf{L}$ in Eq. (3), Eq. (5) immediately means the following equation:

$$\begin{aligned} & E_{i+1,0} \circ \oplus_{\varepsilon_{i,0}} \circ \big(B_1 \circ \mathbf{S}_0^i(\alpha) \oplus B_2 \circ \mathbf{S}_1^i(0) \oplus \\ & B_2 \circ \mathbf{S}_2^i(0) \oplus B_3 \circ \mathbf{S}_3^i(0)\big) \\ = \; & E_{i+1,0} \circ \oplus_{\varepsilon_{i,0}} \circ \big(B_1 \circ \mathbf{S}_0^i(0) \oplus B_2 \circ \mathbf{S}_1^i(\beta) \oplus \\ & B_2 \circ \mathbf{S}_2^i(0) \oplus B_3 \circ \mathbf{S}_3^i(0)\big), \end{aligned}$$

where $\varepsilon_{i,0}$ is the corresponding byte of the constant ε_i. Since $E_{i+1,0}$ is a bijection, we have the following equation:

$$B_1 \circ \mathbf{S}_0^i(\alpha) \oplus B_2 \circ \mathbf{S}_1^i(0) = B_1 \circ \mathbf{S}_0^i(0) \oplus B_2 \circ \mathbf{S}_1^i(\beta).$$

For convenience, define $u_m = \mathbf{S}_0^i(m)$ and $v_m = \mathbf{S}_1^i(m)$, then we have

$$B_1 \circ (u_0 \oplus u_\alpha) = B_2 \circ (v_0 \oplus v_\beta). \tag{6}$$

Since $\alpha \mapsto f_0^i(\alpha,0,0,0)$ and $\beta \mapsto f_0^i(0,\beta,0,0)$ are bijections, we can find 256 collisions. After removing $(\alpha,\beta) = (0,0)$, we get 255 pairs (α,β) satisfying Eq. (5), each providing an equation of the form of Eq. (6). In the same way, we use other f_j^i functions ($j \in \{1,2,3\}$) to generate similar equations with different coefficients in $\{B_1, B_2, B_3\}$. Finally, we get 4×255 linear equations with all 512 unknowns, as follows:

$$\begin{cases} B_1 \circ (u_0 \oplus u_\alpha) = B_2 \circ (v_0 \oplus v_\beta); \\ B_3 \circ (u_0 \oplus u_\alpha) = B_1 \circ (v_0 \oplus v_\beta); \\ B_2 \circ (u_0 \oplus u_\alpha) = B_3 \circ (v_0 \oplus v_\beta); \\ B_2 \circ (u_0 \oplus u_\alpha) = B_2 \circ (v_0 \oplus v_\beta). \end{cases} \tag{7}$$

Define $u'_m = u_0 \oplus u_m$ and $v'_m = v_0 \oplus v_m$, with $m \in \{1,2,\ldots,255\}$, so that the number of unknowns is reduced to $2 \times 255 = 510$. Thus, Eq. (6) can be rewritten as

$$B_1 \circ u'_\alpha = B_2 \circ v'_\beta,$$

meaning that the linear system of Eq. (7) can be represented with 510 unknowns as

$$\begin{cases} B_1 \circ u'_\alpha = B_2 \circ v'_\beta, \\ B_3 \circ u'_\alpha = B_1 \circ v'_\beta, \\ B_2 \circ u'_\alpha = B_3 \circ v'_\beta, \\ B_2 \circ u'_\alpha = B_2 \circ v'_\beta. \end{cases}$$

The 4×255 equations yield a linear system of rank 509; and in such a linear equation system, all other unknowns can be expressed as a function of one of them, say u'_1, that is, there exist coefficients a_i and b_i such that $u'_m = a_m \cdot u'_1$ and $v'_m = b_m \cdot u'_1$. That is,

$$\begin{aligned} u_m &= a_m \cdot (u_0 \oplus u_1) \oplus u_0, \\ v_m &= b_m \cdot (u_0 \oplus u_1) \oplus v_0. \end{aligned} \tag{8}$$

Next we can recover the $\mathbf{S}_0^i$ function by exhaustive search on the pair (u_0,u_1), and at last we use the following equation from the definition of the $\mathbf{S}_0^i$ function to verify whether the obtained $\mathbf{S}_0^i$ function is right or not:

$$\mathbf{S}^{-1} \circ \mathbf{S}_0^i(\cdot) = rk_{i,0} \oplus E_{i,0}^{-1}(\cdot).$$

Since $E_{i,0}^{-1}$ is an 8×8-bit invertible affine transformation, the above function has an algebraic degree of at most 1. For a wrong pair (u_0,u_1), a wrong candidate function $\mathbf{S}_0^{i,*}$ would be got which is an affine equivalent to $\mathbf{S}_0^i$, namely there exists an 8×8-bit matrix a and an 8-bit vector b such that $\mathbf{S}_0^{i,*}(\cdot) = a \cdot \mathbf{S}_0^i(\cdot) \oplus b$, with $a \neq 0$ and $(a,b) \neq (0,1)$. The function $\mathbf{S}^{-1} \circ \mathbf{S}_0^{i,*}(\cdot)$ satisfies

$$\mathbf{S}^{-1} \circ \mathbf{S}_0^{i,*}(\cdot) = \mathbf{S}^{-1}\big(a \cdot \mathbf{S}\big(rk_{i,0} \oplus E_{i,0}^{-1}(\cdot)\big) \oplus b\big).$$

In this case, $\mathbf{S}^{-1} \circ \mathbf{S}_0^{i,*}(\cdot)$ has an algebraic degree greater than 1 with an overwhelming probability. More specifically, we set the function $\hat{g}(\cdot) = \mathbf{S}^{-1} \circ \mathbf{S}^{i,*}(\cdot)$, used Lai's higher-order derivative concept [21] to calculate the first-order derivative of $\hat{g}$, and finally ran ten thousand tests without obtaining a function with an algebraic degree of 1 or less. For instance, the first-order derivative $\hat{\varphi}$ at point $(\mathtt{01})$ is set to

$$\hat{\varphi}(x) = \hat{g}(x \oplus \mathtt{01}) \oplus \hat{g}(x),$$

and we verify whether $\hat{\varphi}(x)$ is constant with at most 2^7 inputs of x, since $\hat{\varphi}(x) = \hat{\varphi}(x \oplus \mathtt{01})$. For each wrong pair, the probability that $\hat{\varphi}(x)$ is constant is roughly 2^{-8}, so wrong guesses can be quickly removed.

After recovering $\mathbf{S}_0^i$, we can use Eq. (8) to recover $\mathbf{S}_1^i$ by exhaustive search on v_0, and similarly recover $\mathbf{S}_2^i$ and $\mathbf{S}_3^i$ with other equations finally.

3) Recovering the Linear Part of Encoding E_{i+1}: After the $\mathbf{S}_j^i$ functions have been recovered ($j = 0,1,2,3$), however it is not as easy to recover the output encodings $E_{i+1,j}$ as Lepoint et al.'s attack on Chow et al.'s white-box AES implementation, because of the existence of the unknown constant ε_i, which is partially due to the different structures of Feistel and SPN ciphers and the design of Yao and Chen's white-box SM4 implementation. Anyway, we find a trick to recover the linear part of the output encodings $E_{i+1,j}$. Since $E_{i+1,j}$ is an invertible affine transformation, we write

$E_{i+1,j}(\cdot) = C_{i+1,j}(\cdot) \oplus c_{i+1,j}$, where the linear part $C_{i+1,j}$ is a general invertible 8×8-bit matrix and $c_{i+1,j}$ is an 8-bit constant.

Given a 32-bit input $Z_i = (z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3})$ to the f^i collision function, denote the original 32-bit value immediately after the $\widehat{\mathbf{L}}$ operation as follows:

$$\begin{aligned} W_i &= \begin{bmatrix} W_{i,0} & W_{i,1} & W_{i,2} & W_{i,3} \end{bmatrix}^T \\ &= \mathbf{L}_0 \circ \mathbf{S}_0^i(z_{i,0}) \oplus \mathbf{L}_1 \circ \mathbf{S}_1^i(z_{i,1}) \oplus \\ &\quad \mathbf{L}_2 \circ \mathbf{S}_2^i(z_{i,2}) \oplus \mathbf{L}_3 \circ \mathbf{S}_3^i(z_{i,3}). \end{aligned}$$

As $\mathbf{L}$ is public and we have recovered $\mathbf{S}_j^i$ above ($j = 0, 1, 2, 3$), we can compute Y_j. The output of the f^i collision function is

$$f^i = \begin{bmatrix} f_0^i & f_1^i & f_2^i & f_3^i \end{bmatrix}^T = \begin{bmatrix} E_{i+1,0}(W_{i,0} \oplus \varepsilon_{i,0}) \\ E_{i+1,1}(W_{i,1} \oplus \varepsilon_{i,1}) \\ E_{i+1,2}(W_{i,2} \oplus \varepsilon_{i,2}) \\ E_{i+1,3}(W_{i,3} \oplus \varepsilon_{i,3}) \end{bmatrix},$$

where $(\varepsilon_{i,0}, \varepsilon_{i,1}, \varepsilon_{i,2}, \varepsilon_{i,3}) = \varepsilon_i$.

Subsequently, to recover $E_{i+1,j}$, we need to know the 8-bit unknown constant $\varepsilon_{i,j}$. A straightforward way is to try by exhaustive search, which would cause an additional complexity factor of 2^8. However, we find we can recover the linear part $C_{i+1,j}$ at ease with a negligible time complexity, as follows.

First, we consider the output of the arbitrary 32-bit input $Z_i = (z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3})$ under the f_0^i collision function,

$$\begin{aligned} f_0^i(Z_i) &= E_{i+1,0}(W_{i,0} \oplus \varepsilon_{i,0}) \\ &= C_{i+1,0}(W_{i,0}) \oplus C_{i+1,0}(\varepsilon_{i,0}) \oplus c_{i+1,0}, \end{aligned} \quad (9)$$

where $W_{i,0}$ is defined above, which denotes the corresponding original 8-bit value immediately after the $\mathbf{L}$ operation under the input X.

Next, we choose the 32-bit input $Z0 = (\hat{z}_{i,0}, \hat{z}_{i,1}, \hat{z}_{i,2}, \hat{z}_{i,3})$ to the f^i collision function, so that the original 32-bit value immediately after the $\mathbf{L}$ operation is 0; this can be done easily by choosing $Z0$ such that

$$(\mathbf{S}_0^i(\hat{z}_{i,0}), \mathbf{S}_1^i(\hat{z}_{i,1}), \mathbf{S}_2^i(\hat{z}_{i,2}), \mathbf{S}_3^i(\hat{z}_{i,3})) = \mathbf{L}^{-1}(0) = 0.$$

Thus, its corresponding output under the f_0^i collision function is

$$f_0^i(X0) = C_{i+1,0}(\varepsilon_{i,0}) \oplus c_{i+1,0}. \quad (10)$$

At last, XORing Eq. (9) and Eq. (10), we get $f_0^i(Z_i) \oplus f_0^i(Z0) = C_{i+1,0}(W_{i,0})$. As a consequence, we can recover the linear part $C_{i+1,0}$ of the output encodings $E_{i+1,0}$. The linear parts of other output encodings $E_{i+1,j}$ can be recovered similarly.

4) Recovering the Masked Round Key $rk_{i+1} \oplus C_{i+1}^{-1}(c_{i+1})$: We similarly define the collision function f^{i+1} starting from the $(i+1)$-th round. Subsequently, as Lepoint et al. did on Chow et al.'s white-box AES implementation, we would target to recover the round key rk_{i+1} in the following $(i+1)$-th round, but nevertheless we cannot recover a round key byte from the collision function, because $\varepsilon_{i+1,0}$ and $E_{i+1,j}$ are unknown, although the linear part $C_{i+1,j}$ can be recovered as above. At present we can only recover the masked round key $rk_{i+1} \oplus C_{i+1}^{-1}(c_{i+1})$, as follows.

Based on the equations similar to Eq. (4) and Eq. (5), we define

$$\begin{aligned} g(x) &= f_0^{i+1}\big(E_{i+1,0}(\mathbf{S}^{-1}(x) \oplus rk_{i+1,0}), 0, 0, 0\big) \\ &= f_0^{i+1}\big(C_{i+1,0}(\mathbf{S}^{-1}(x)) \oplus E_{i+1,0}(rk_{i+1,0}), 0, 0, 0\big) \\ &= E_{i+2,0}(B_1(x) \oplus \delta \oplus \varepsilon_{i+1,0}), \end{aligned}$$

where $\delta = B_2 \circ \mathbf{S}_1^{i+1}(0) \oplus B_2 \circ \mathbf{S}_2^{i+1}(0) \oplus B_3 \circ \mathbf{S}_3^{i+1}(0)$ is a constant that can be easily computed.

Because $E_{i+2,0}$ is an 8×8-bit affine transformation, the function g has an algebraic degree of at most 1. For a wrong guess $E_{i+1,0}(\hat{rk}_{i+1,0}) \neq E_{i+1,0}(rk_{i+1,0})$, the function $\hat{g}$ is defined as

$$\begin{aligned} \hat{g}(x) &= f_0^{i+1}\big(C_{i+1,0}(\mathbf{S}^{-1}(x)) \oplus E_{i+1,0}(\hat{rk}_{i+1,0}), 0, 0, 0\big) \\ &= E_{i+2,0}(B_1 \circ \mathbf{S}(\mathbf{S}^{-1}(x) \oplus \hat{rk}_{i+1,0} \oplus rk_{i+1,0}) \oplus \delta \oplus \\ &\quad \varepsilon_{i+1,0}). \end{aligned}$$

In this case, with a similar test, $\hat{g}$ has an algebraic degree of more than 1 with an overwhelming probability. We extract $E_{i+1,0}(rk_{i+1,0})$ by exhaustive search, that is, similarly we verify whether the first-order derivative $\hat{\varphi}(x) = \hat{g}(x \oplus \mathtt{01}) \oplus \hat{g}(x)$ of $\hat{g}(x)$ at point $\mathtt{01}$ is constant for each guess $E_{i+1,0}(\hat{rk}_{i+1,0})$. For a wrong guess $E_{i+1,0}(\hat{rk}_{i+1,0})$, the probability that $\hat{\varphi}(x)$ is constant is roughly 2^{-8}, so wrong guesses can be quickly removed.

As a result, we can also recover $E_{i+1,j}(rk_{i+1,j})$ for $j = 1, 2, 3$, by changing the definition of the function g. Since $E_{i+1}(rk_{i+1}) = C_{i+1}(rk_{i+1}) \oplus c_{i+1} = C_{i+1}(rk_{i+1} \oplus C_{i+1}^{-1}(c_{i+1}))$, we can get the masked round key $rk_{i+1} \oplus C_{i+1}^{-1}(c_{i+1})$, where $C_{i+1}^{-1}(c_{i+1})$ is an unknown Boolean mask.

5) Recovering Encodings P_{i+4}, P'_{i+4} and P''_{i+4} Partially or Completely: After recovering $\mathbf{S}_0^i$, $\mathbf{S}_1^i$, $\mathbf{S}_2^i$ and $\mathbf{S}_3^i$ functions in the first phase, we can easily recover the general 32×32-bit affine encoding P''_{i+4} used in the i-th round, see Fig. 2 or Fig. 4, since $Y'_{i0} = P''_{i+4}(\mathbf{L} \circ (\mathbf{S}_0^i(z_{i,0}) || \mathbf{S}_1^i(z_{i,1}) || \mathbf{S}_2^i(z_{i,2}) || \mathbf{S}_3^i(z_{i,3})))$.

Even further, recall that $P_i(x) = A_i \cdot x \oplus a_i$, $P'_{i+4}(x) = P_{i+4} \oplus a'_{i+4}$ and $P''_{i+4}(x) = P_{i+4} \oplus a''_{i+4}$, we can learn that $a_i = a'_{i+4} \oplus a''_{i+4}$, and consequently we can recover both the linear part A_{i+4} and the constant part a'_{i+4} of P''_{i+4}, as follows. First, we recover the linear part A_{i+4} in a way similar to recovering the linear part of encoding E_{i+1} above, that is, by considering the output difference under a pair of inputs:

$$\begin{aligned} & Y'_{i0} \oplus \hat{Y}'_{i0} \\ = \; & P''_{i+4}(\mathbf{L} \circ (\mathbf{S}_0^i(z_{i,0}) || \mathbf{S}_1^i(z_{i,1}) || \mathbf{S}_2^i(z_{i,2}) || \mathbf{S}_3^i(z_{i,3}))) \oplus \\ & P''_{i+4}(\mathbf{L} \circ (\mathbf{S}_0^i(\hat{z}_{i,0}) || \mathbf{S}_1^i(\hat{z}_{i,1}) || \mathbf{S}_2^i(\hat{z}_{i,2}) || \mathbf{S}_3^i(\hat{z}_{i,3}))) \\ = \; & A_{i+4}(\mathbf{L} \circ (\mathbf{S}_0^i(z_{i,0}) || \mathbf{S}_1^i(z_{i,1}) || \mathbf{S}_2^i(z_{i,2}) || \mathbf{S}_3^i(z_{i,3})) \oplus \\ & \mathbf{L} \circ (\mathbf{S}_0^i(\hat{z}_{i,0}) || \mathbf{S}_1^i(\hat{z}_{i,1}) || \mathbf{S}_2^i(\hat{z}_{i,2}) || \mathbf{S}_3^i(\hat{z}_{i,3}))). \end{aligned}$$

The constant part a'_{i+4} can be easily obtained, as we have $Y'_{i0} = P''_{i+4}(\mathbf{L} \circ (\mathbf{S}_0^i(z_{i,0}) || \mathbf{S}_1^i(z_{i,1}) || \mathbf{S}_2^i(z_{i,2}) || \mathbf{S}_3^i(z_{i,3}))) = A_{i+4}(\mathbf{L} \circ (\mathbf{S}_0^i(z_{i,0}) || \mathbf{S}_1^i(z_{i,1}) || \mathbf{S}_2^i(z_{i,2}) || \mathbf{S}_3^i(z_{i,3}))) \oplus a'_{i+4}$.

As given above, because P_{i+4} and P'_{i+4} use the same linear part A_{i+4} as P''_{i+4}, we can also know the linear part A_{i+4} of P_{i+4} or P'_{i+4}. (But it seems impossible to recover a_{i+4} and a''_{i+4}.)

6) Time Complexity: In the phase of recovering $\mathbf{S}_0^i$, there are 2^{16} candidates (u_0, u_1) for exhaustive search, and to verify whether $\hat{\varphi}(x)$ is constant we need to calculate $\hat{\varphi}(x)$ for at most 2^7 inputs. For a wrong guess (u_0, u_1), the probability that $\hat{\varphi}(x)$ is constant is 2^{-8} roughly. Thus, the expected value of the test is $1+1/256+\cdots+1/(256^{127}) \approx 1$. The expected time complexity of recovering $\mathbf{S}_0^i$ is hence about $2^{16} \cdot 1 \cdot 2 = 2^{17}$ (dominated by $\mathbf{S}/\mathbf{S}^{-1}$ computations). We recover $\mathbf{S}_1^i$, $\mathbf{S}_2^i$ and $\mathbf{S}_3^i$ by exhaustive search on v_0 and produce an expected time complexity of $3 \cdot (2^8 \cdot 1 \cdot 2) = 3 \cdot 2^9$. Thus, the expected time complexity of recovering all the four $\mathbf{S}_j^i$'s is about $2^{17}+3\cdot 2^9 = 259 \cdot 2^9$.

The time complexity for recovering the linear part of output encoding $E_{i+1,j}$ is negligible. The expected time complexity of recovering $E_{i+1,0}(rk_{i+1,0})$ is about $2^8 \cdot 1 \cdot 2 = 2^9$, so the total expected time complexity of recovering $E_{i+1}(rk_{i+1})$ is about $259 \cdot 2^9 + 4 \cdot (2^8 \cdot 1 \cdot 2) = 259 \cdot 2^9 + 2^{11}$, and the expected time complexity is about $31 \times (259 \cdot 2^9 + 2^{11}) \approx 2^{22}$ to recover all the masked round keys $rk_{i+1} \oplus C_{i+1}^{-1}(c_{i+1})$ in the last 31 rounds $(i = 1, 2, \cdots, 31)$.

7) A Somewhat Equivalent White-Box SM4 Implementation Mostly with Boolean Masks: After we recover both the linear and constant parts of encoding P''_{i+4} and the linear parts of encodings P_{i+4}, P'_{i+4} and E_{i+1} above $(i = 0, 1, \cdots, 30)$, we can peel off the affine encodings P''_{i+4} and the linear parts A_{i+4} and C_{i+1} of the affine encodings P_{i+4}, P'_{i+4} and E_{i+1} in the 32 rounds, and thus it is easy to see that we can simplify Yao and Chen's white-box SM4 implementation into such an equivalent white-box SM4 implementation that the 32 rounds have only Boolean masks for the encodings except P_0, P_1, P_2, P_3, E_0 and P_{35}; in particular, each X_{i+4} is masked by an unknown Boolean constant, and the $(i+1)$-th round transformation $\mathbf{T}$ has masked input but original output. Note that by "somewhat equivalent" we mean this simplifying process still has a workload, and thus the simplifying process is costly when one would like to have a security expectation below 2^{22}, but it is desirable on the contrary, since the cumbersome affine encodings are not required to generate any longer from the security's perspective.

So far we can only recover the masked round keys $rk_{i+1} \oplus C_{i+1}^{-1}(c_{i+1})$, and it seems impossible to work out the original key further, and alternatively one may prove this is impossible further. We leave it as an open problem.

IV. Cryptanalysis of Xiao and Lai's White-Box SM4 Implementation

Xiao and Lai's white-box SM4 implementation [38] is similar to Yao and Chen's white-box SM4 implementation at a high level, except that there is no state expansion to the S-box layer and thus the original $\mathbf{L}$ operation is used. Fig. 5 depicts an encryption round of Xiao and Lai's white-box SM4 implementation, where Q_i is a general invertible affine output encoding. Therefore, we can apply our above cryptanalysis to Xiao and Lai's white-box SM4 implementation in the same way, and the total expected time complexity is also about 2^{22} for recovering recover all the masked round keys in the last 31 rounds. Likewise, we can peel off most white-box operations and make a somewhat equivalent white-box SM4 implementation mostly with Boolean masks.

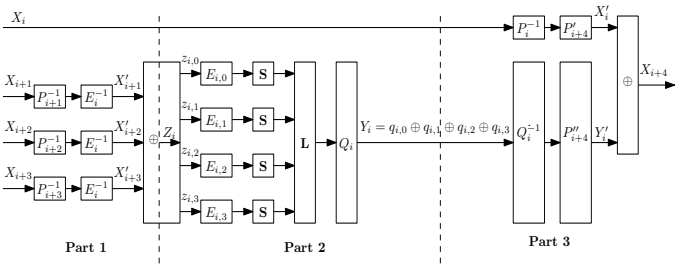

Figure 5. An encryption round of Xiao and Lai's white-box SM4 implementation

V. Cryptanalysis of Shang's White-Box SM4 Implementation

Shang's white-box SM4 implementation [31] is based on Xiao and Lai's white-box SM4 implementation, mainly by applying two general 16-bit affine encodings $E_{i,0}$ and $E_{i,1}$ to the input of the S-box layer in parallel, each corresponding to two S-boxes and subsequently a 32×16-bit component $\mathbf{L}_0$ or $\mathbf{L}_1$ of the $\mathbf{L}$ matrix, and thus two 16×32-bit tables. Fig. 6 depicts an encryption round of Shang's white-box SM4 implementation.

We can similarly exploit our above collision-based attack to Shang's white-box SM4 implementation after a few modifications, to recover some masked round keys and peel off most white-box operations to a somewhat equivalent implementation. Specifically, we represent $\mathbf{L}$ by Eq. (3) with two 16×16-bit blocks $\hat{L}_0$ and $\hat{L}_1$ as

$$\mathbf{L} = \begin{bmatrix} \hat{L}_0 & \hat{L}_1 \\ \hat{L}_1 & \hat{L}_0 \end{bmatrix},$$

define

$$\mathbf{S}_0^i(x) = \begin{pmatrix} \mathbf{S} \\ \mathbf{S} \end{pmatrix} \big((rk_{i,0}||rk_{i,1}) \oplus E_{i,0}(x)\big),$$
$$\mathbf{S}_1^i(x) = \begin{pmatrix} \mathbf{S} \\ \mathbf{S} \end{pmatrix} \big((rk_{i,2}||rk_{i,3}) \oplus E_{i,1}(x)\big),$$

where $x \in \mathrm{GF}(2)^{16}$, and define a collision function on Shang's white-box SM4 implementation whose equivalent f^i

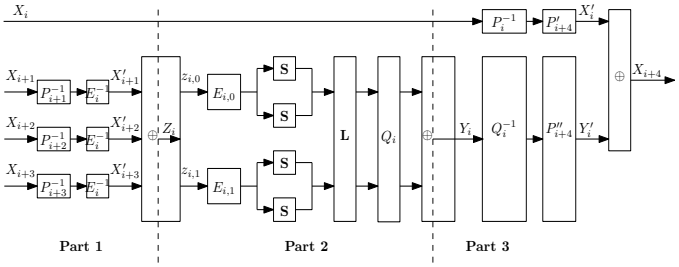

Figure 6. An encryption round of Shang's white-box SM4 implementation

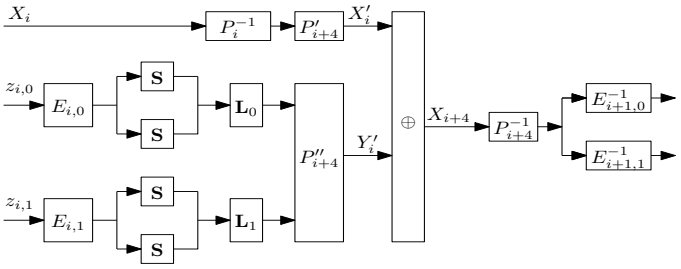

Figure 7. Equivalent of collision function on Shang's white-box SM4 implementation

is depicted in Fig. 7, as follows:

$$\begin{aligned} & f^i(z_{i,0}, z_{i,1}) = [f_0^i(z_{i,0}, z_{i,1}), f_1^i(z_{i,0}, z_{i,1})]^T \\ &= \begin{bmatrix} E_{i+1,0}^{-1} \\ E_{i+1,1}^{-1} \end{bmatrix} \oplus_{\epsilon_i} \circ \mathbf{L} \circ \begin{bmatrix} \begin{pmatrix} \mathbf{S} \\ \mathbf{S} \end{pmatrix} \left((rk_{i,0}||rk_{i,1}) \oplus E_{i,0}(z_{i,0})\right) \\ \begin{pmatrix} \mathbf{S} \\ \mathbf{S} \end{pmatrix} \left((rk_{i,2}||rk_{i,3}) \oplus E_{i,1}(z_{i,1})\right) \end{bmatrix} \\ &= \begin{bmatrix} E_{i+1,0}^{-1} \\ E_{i+1,1}^{-1} \end{bmatrix} \oplus_{\epsilon_i} \circ \mathbf{L} \circ \begin{bmatrix} \mathbf{S}_0^i(z_{i,0}) \\ \mathbf{S}_1^i(z_{i,1}) \end{bmatrix}. \end{aligned}$$

Then, we consider the collision $f_h^i(\alpha, 0) = f_h^i(0, \beta)$, where $\alpha, \beta \in \text{GF}(2)^{16}$ and $h = 0, 1$. At last, we get the following linear system of $2 \times (2^{16} - 1)$ equations with $2 \times (2^{16} - 1)$ unknowns:

$$\begin{aligned} \hat{L}_0 \circ u'_\alpha \oplus \hat{L}_1 \circ v'_\beta = 0, \\ \hat{L}_1 \circ u'_\alpha \oplus \hat{L}_0 \circ v'_\beta = 0, \end{aligned}$$

where $u'_\alpha = \mathbf{S}_0^i(\alpha) \oplus \mathbf{S}_0^i(0)$ and $v'_\beta = \mathbf{S}_1^i(\beta) \oplus \mathbf{S}_1^i(0)$ and $(\alpha, \beta) \neq (0, 0)$. Subsequently, by a similar process, we can recover the $\mathbf{S}_h^i$ function with an expected time complexity of about $2^{32} \cdot 1 \cdot 2 \cdot 2 + 2^{16} \cdot 1 \cdot 2 \cdot 2 = 2^{34} + 2^{18}$, and recover the linear part of $E_{i+1,h}$ with a negligible time complexity. Note that here each $\mathbf{S}_h^i$ computation involves two $\mathbf{S}$ computations.

At last, suppose that $E_{i+1,h}(\cdot) = C_{i+1,h}(\cdot) \oplus c_{i+1,h}$ and the linear part $C_{i+1,h}$ has been recovered as above, where $C_{i+1,h}$ is a general invertible 16×16-bit matrix and $c_{i+1,h}$ is a 16-bit constant. Similarly, we depend on the following function to recover the masked key bytes $(rk_{i+1,0}||rk_{i+1,1}) \oplus C_{i+1,h}^{-1}(\oplus c_{i+1,h})$ with an expected time complexity of about $2^{16} \cdot 1 \cdot 2 \cdot 2 = 2^{18}$:

$$\begin{aligned} & f_0^{i+1}\big(E_{i+1,0}^{-1}\big(\begin{pmatrix} \mathbf{S}^{-1} \\ \mathbf{S}^{-1} \end{pmatrix}(x) \oplus (rk_{i+1,0}||rk_{i+1,1})\big), 0\big) \\ &= f_0^{i+1}\big(C_{i+1,0}^{-1}\big(\begin{pmatrix} \mathbf{S}^{-1} \\ \mathbf{S}^{-1} \end{pmatrix}(x)\big) \oplus E_{i+1,0}^{-1}(rk_{i+1,0}||rk_{i+1,1}), 0\big) \\ &= E_{i+2,0}^{-1}(L_0(x) \oplus L_1 \circ \mathbf{S}_1^{i+1}(0) \oplus \varepsilon_{i+1,0}), \end{aligned}$$

where $\varepsilon_{i+1,0}$ is the corresponding 16-bit part of ε_{i+1}.

Thus, the total expected time complexity is about $2^{34} + 2^{18} + 2 \cdot 2^{18} \approx 2^{34}$ for recovering the masked round key $rk_{i+1} \oplus C_{i+1}^{-1}(\oplus c_{i+1})$ from Shang's white-box SM4 implementation, and the expected time complexity is about $31 \times 2^{34} \approx 2^{39}$ to recover all the masked round keys $rk_{i+1} \oplus C_{i+1}^{-1}(\oplus c_{i+1})$ in the last 31 rounds $(i = 1, 2, \cdots, 31)$. As a consequence, we can similarly peel off most white-box operations and make a somewhat equivalent white-box SM4 implementation mostly with Boolean masks.

VI. Collision-Based Attack on Wu et al.'s White-Box SM4 Implementation

In this section, we briefly describe Wu et al.'s white-box SM4 implementation and our attack.

A. Wu et al.'s White-Box SM4 Implementation

An encryption round of Wu et al.'s white-box SM4 implementation [35] is made up of three parts, as depicted in Fig. 8, but it is expanded to 36 rounds to produce the original output (without protection), and there are respectively two types of lookup tables in the second and third parts, especially, the second type of lookup tables uses three different construction methods for different rounds.

1) Part 1: The first part is processed as follows, and the 32-bit output Z_i is protected by a diagonal invertible matrix E_i:

$$\begin{aligned} X'_{i+j} &= A_{i,j}(X_{i+j}), j = 1, 2, 3; \\ Z_i &= X'_{i+1} \oplus X'_{i+2} \oplus X'_{i+3}, \end{aligned}$$

where $A_{i,j}$ is a composite 32-bit invertible matrix with different encodings $(i = 0, 1, \ldots, 35)$, as follows,

$$\begin{array}{ccc} A_{0,1} = E_0 P, & A_{0,2} = E_0 P, & A_{0,3} = E_0 P, \\ A_{1,1} = E_1 P, & A_{1,2} = E_1 P, & A_{1,3} = E_1 R_0^{-1}, \\ A_{2,1} = E_2 P, & A_{2,2} = E_2 R_0^{-1}, & A_{2,3} = E_2 R_1^{-1}, \\ A_{3,1} = E_3 R_0^{-1}, & A_{3,2} = E_3 R_1^{-1}, & A_{3,3} = E_3 R_2^{-1}, \\ A_{4,1} = E_4 R_1^{-1}, & A_{4,2} = E_4 R_2^{-1}, & A_{4,3} = E_4 R_3^{-1}, \\ \vdots & \vdots & \vdots \\ A_{31,1} = E_{31} R_{28}^{-1}, & A_{31,2} = E_{31} R_{29}^{-1}, & A_{31,3} = E_{31} R_{30}^{-1}, \\ A_{32,1} = E_{32} R_{25}^{-1}, & A_{32,2} = E_{32} R_{26}^{-1}, & A_{32,3} = E_{32} R_{27}^{-1}, \\ A_{33,1} = E_{33} R_{26}^{-1}, & A_{33,2} = E_{33} R_{27}^{-1}, & A_{33,3} = E_{33} R_{28}^{-1}, \\ A_{34,1} = E_{34} R_{27}^{-1}, & A_{34,2} = E_{34} R_{28}^{-1}, & A_{34,3} = E_{34} R_{29}^{-1}, \\ A_{35,1} = E_{35} R_{28}^{-1}, & A_{35,2} = E_{35} R_{29}^{-1}, & A_{35,3} = E_{35} R_{30}^{-1}, \end{array}$$

with $P = \text{diag}(P_0, P_1, P_2, P_3)$ being a diagonal invertible matrix and $R_0, R_1, \cdots, R_{30}$ being general invertible matrices.

2) Part 2: Construct four 8-bit to 32-bit lookup tables of the first type $Table_{i,j}$ $(j = 0, 1, 2, 3)$, and XOR the outputs of the four tables into a 32-bit Y_i, as follows:

$$\begin{aligned} Y_i &= \bigoplus_{j=0}^{3} Table_{i,j} \\ &= Q_i \circ P \circ \mathbf{L} \circ \begin{bmatrix} \mathbf{S} \circ \oplus_{rk_{i,0}} \circ P_0^{-1} \circ E_{i,0}^{-1}(z_{i,0}) \\ \mathbf{S} \circ \oplus_{rk_{i,1}} \circ P_1^{-1} \circ E_{i,1}^{-1}(z_{i,1}) \\ \mathbf{S} \circ \oplus_{rk_{i,2}} \circ P_2^{-1} \circ E_{i,2}^{-1}(z_{i,2}) \\ \mathbf{S} \circ \oplus_{rk_{i,3}} \circ P_3^{-1} \circ E_{i,3}^{-1}(z_{i,3}) \end{bmatrix}. \end{aligned}$$

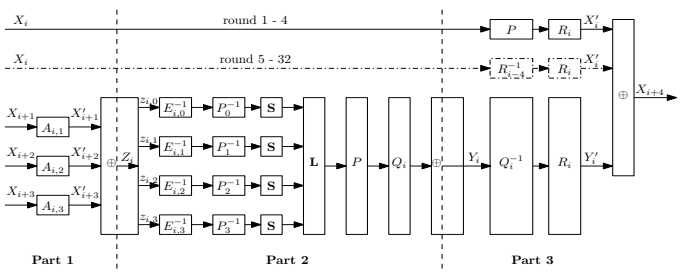

Figure 8. An encryption round of Wu et al.'s white-box SM4 implementation

3) Part 3: Construct four 16-bit to 8-bit lookup tables of the second type. Let (X_i, Y_i) be the input of the four tables, and X_{i+4} be the output, then

- Rounds 1-4:

$$X_i' = R_i \circ P(X_i), Y_i' = R_i \circ Q_i^{-1}(Y_i),$$
$$X_{i+4} = Y_i' \oplus X_i';$$

- Rounds 5-32:

$$X_i' = R_i \circ R_{i-4}(X_i), Y_i' = R_i \circ Q_i^{-1}(Y_i),$$
$$X_{i+4} = Y_i' \oplus X_i';$$

- Rounds 33-36:

$$X_i' = R_i \circ R_{i-8}(X_i), Y_i' = P^{-1} \circ Q_i^{-1}(Y_i),$$
$$X_{i+4} = Y_i' \oplus X_i'.$$

B. Attacking Wu et al.'s Implementation

Note that all the encodings are linear (invertible matrices) in Wu et al.'s white-box SM4 implementation, rather than affine encodings as in the above three white-box SM4 implementations. As a consequence, we can devise a collision function in a similar way as above, but much easier, since there is no effect of unknown constant parts associated with the encodings. Fig. 9 (top) depicts a collision function on Wu et al.'s white-box SM4 implementation, where Q_i and Q_i^{-1} are cancelled with each other.

Further, the collision function can be simplified. First, we set $X_i = 0$, and thus $R_i \circ P(X_i) = 0$, since R_i and P are invertible matrices. Second, R_i and R_i^{-1} are cancelled with each other. Thus, after we adjust the position of encoding P, we get a simplified collision function as depicted in Fig. 9 (bottom), that is

$$\begin{aligned} & f(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}) \\ =& [f_0(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}), f_1(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}), \\ & f_2(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3}), f_3(z_{i,0}, z_{i,1}, z_{i,2}, z_{i,3})]^T \\ =& \begin{bmatrix} \hat{E}_{i+1,0} \\ \hat{E}_{i+1,1} \\ \hat{E}_{i+1,2} \\ \hat{E}_{i+1,3} \end{bmatrix} \circ \mathbf{L} \circ \begin{bmatrix} \mathbf{S} \circ \oplus_{rk_{i,0}} \circ \hat{E}_{i,0}^{-1}(z_{i,0}) \\ \mathbf{S} \circ \oplus_{rk_{i,1}} \circ \hat{E}_{i,1}^{-1}(z_{i,1}) \\ \mathbf{S} \circ \oplus_{rk_{i,2}} \circ \hat{E}_{i,2}^{-1}(z_{i,2}) \\ \mathbf{S} \circ \oplus_{rk_{i,3}} \circ \hat{E}_{i,3}^{-1}(z_{i,3}) \end{bmatrix}, \end{aligned}$$

where $\hat{E}_{i,j}^{-1}(\cdot) = P_j^{-1} \circ E_{i,j}^{-1}(\cdot)$ and $\hat{E}_{i+1,j}(\cdot) = E_{i+1,j} \circ P_j(\cdot)$ are invertible 8×8-bit matrices ($j = 0, 1, 2, 3$). Note that $\hat{E}_{i,j}^{-1}$ and $\hat{E}_{i+1,j}$ are matrixes simply, not affine transformations, and thus this attack is a simplified case.

Define $\mathbf{S}_j$ function as

$$\begin{aligned} \mathbf{S}_j(\cdot) &= \mathbf{S} \circ \oplus_{rk_{i,j}} \circ \hat{E}_{i,j}^{-1}(\cdot) \\ &= \mathbf{S}(rk_{i,j} \oplus \hat{E}_{i,j}^{-1}(\cdot)), \quad j = 0, 1, 2, 3. \end{aligned}$$

Then, we can similarly recover the four functions $\mathbf{S}_j$'s with an expected time complexity of about $2^{16} \cdot 1 \cdot 2 + 3 \cdot (2^8 \cdot 1 \cdot 2) = 259 \cdot 2^9$. Since $\hat{E}_{i+1,0}$ is an invertible 8×8-bit matrix, we can recover it immediately by calculating $\hat{E}_{i+1,0}(\cdot) = f_0(\psi^{-1}(\cdot), 0, 0, 0)$, where $\psi : \alpha \mapsto B_1 \circ \mathbf{S}_0(\alpha) \oplus B_2 \circ \mathbf{S}_1(0) \oplus B_2 \circ \mathbf{S}_2(0) \oplus B_3 \circ \mathbf{S}_3(0)$. Similarly for recovering $\hat{E}_{i+1,1}, \hat{E}_{i+1,2}$ and $\hat{E}_{i+1,3}$.

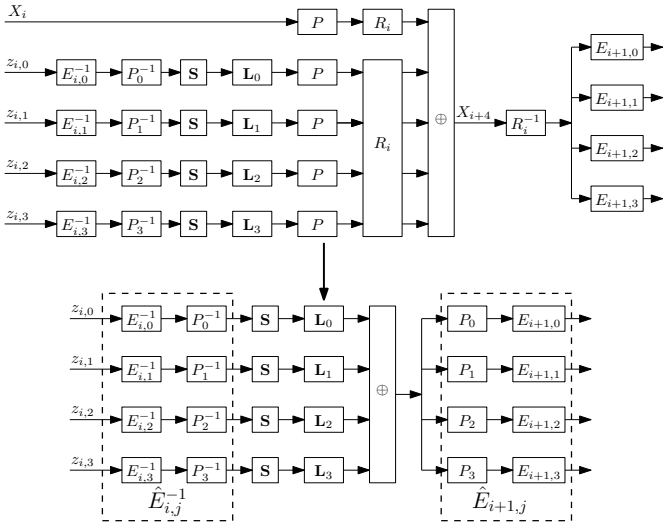

Figure 9. Collision function and its equivalent on Wu et al's white-box SM4 implementation

At last, we set function g as

$$\begin{aligned} g(x) =& f_j\big(\hat{E}_{i,0}(\mathbf{S}^{-1}(x) \oplus rk_{i,0}), 0, 0, 0\big) \\ =& \hat{E}_{i+1,0}(B_1 \circ x \oplus B_2 \circ \mathbf{S}_1(0) \oplus B_2 \circ \mathbf{S}_2(0) \oplus \\ & B_3 \circ \mathbf{S}_3(0)), \end{aligned}$$

and we can similarly recover the round key byte $rk_{i,0}$ and finally the whole round key rk_i with an expected time complexity of about $4 \cdot (2^8 \cdot 1 \cdot 2) = 2^{11}$. Therefore, the total expected time complexity is about $259 \cdot 2^9 + 2^{11} \approx 2^{17.1}$ for recovering a round key.

At last, note that producing unprotected ciphertexts would make Wu et al.'s white-box SM4 implementation be potentially vulnerable to other attacks, see [8].

VII. Concluding Remarks

The SM4 block cipher is a Chinese national standard and an ISO international standard, formerly known as SMS4, and a few white-box SM4 implementations have been proposed with its increasingly wide use. In this paper, we have analysed the security of a type of white-box SM4 constructions that use an affine diagonal block encoding to protect the original XOR sum of the three branches entering the S-box layer of a round and use its inverse to protect the original input of the S-box layer, and have shown that it can be somewhat equivalent to a plain implementation mostly with Boolean masks for the affine case and is insecure for the linear case, by devising collision-based attacks to peel off most white-box operations and recover a masked or original round key. Our cryptanalysis results show that generating such a white-box SM4 implementation with affine encodings can be simplified into generating a plain implementation with Boolean masks (if its security expectation is beyond the given complexity), avoiding the cumbersome affine encodings, and the effect of an affine encoding is significantly better than the effect of a linear encoding in the sense of security. Currently we think it seems impossible to recover the original key from the masked round keys, and leave it as an open problem.

ACKNOWLEDGMENTS

This is an extended version of the paper appeared in Proceedings of ISC 2021 — The 24th Information Security Conference [26]. In this extended version, we revised and corrected our previous cryptanalysis results and conclusions on Yao and Chen's and Xiao and Lai's white-box SM4 implementations, and gave cryptanalysis results on two other white-box SM4 implementations, namely Shang's and Wu et al.'s implementations. The authors are very grateful to Prof. Zheng Gong for pointing out a mistake in an earlier version of this paper. Jiqiang Lu is Qianjiang Special Expert of Hangzhou.

REFERENCES

[1] C. H. Baek, J. H. Cheon, and H. Hong, "White-box AES implementation revisited," *Journal of Communications and Networks*, vol. 18, no. 3, pp. 273–287, 2016.

[2] K. P. Bai and C. K. Wu, "A secure white-box SM4 implementation," *Security and Communication Networks*, vol. 9, no. 10, pp. 996–1006, 2016.

[3] K. P. Bai, C. K. Wu, and Z. F. Zhang, "Protect white-box AES to resist table composition attacks," *IET Information Security*, vol. 12, no. 4, pp. 305–313, 2018.

[4] E. Barkan and E. Biham, "In how many ways can you write Rijndael?," in *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2002)*, LNCS, vol. 2501, pp. 160–175, Springer, 2002.

[5] E. Biham, A. Shamir, Differential cryptanalysis of the Data Encryption Standard. Springer-Verlag, 1993.

[6] O. Billet, H. Gilbert, and C. Ech-Chatbi, "Cryptanalysis of a White Box AES Implementation," in *International Conference on Selected Areas in Cryptography (SAC 2004)*, LNCS, vol. 3357, pp. 227–240, Springer, 2004.

[7] A. Biryukov, C. De Cannière, A. Braeken, and B. Preneel, "A toolbox for cryptanalysis: Linear and affine equivalence algorithms," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2003)*, LNCS, vol. 2656, pp. 33–50, Springer, 2003.

[8] J. W. Bos, C. Hubain, W. Michiels, and P. Teuwen, "Differential computation analysis: Hiding your white-box designs is not enough," in *International Conference on Cryptographic Hardware and Embedded Systems (CHES 2016)*, LNCS, vol. 9813, pp. 215–236, Springer, 2016.

[9] J. Bringer, H. Chabanne, and E. Dottax, "White box cryptography: Another attempt," *IACR Cryptology ePrint Archive*, no. 2006, p. 468, 2006.

[10] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "White-box cryptography and an AES implementation," in *International Conference on Selected Areas in Cryptography (SAC 2002)*, LNCS, vol. 2595, pp. 250–270, Springer, 2002.

[11] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "A white-box DES implementation for DRM applications," in *ACM Workshop on Digital Rights Management (DRM 2002)*, LNCS, vol. 2696, pp. 1–15, Springer, 2002.

[12] Y. De Mulder, P. Roelse, and B. Preneel, "Cryptanalysis of the Xiao-Lai white-box AES implementation," in *International Conference on Selected Areas in Cryptography (SAC 2012)*, LNCS, vol. 7707, no. 1, pp. 34–49, Springer, 2013.

[13] Y. De Mulder, B. Wyseur, and B. Preneel, "Cryptanalysis of a perturbated white-box AES implementation," in *International Conference on Cryptology in India (INDOCRYPT 2010)*, LNCS, vol. 6498, pp. 292–310, Springer, 2010.

[14] P. Derbez, P. A. Fouque, B. Lambin, and B. Minaud, "On recovering affine encodings in white-box implementations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, LNCS, vol. 2018, no. 3, pp. 121–149, 2018.

[15] *The SMS4 Block Cipher (in Chinese)*. Office of State Commercial Cryptography Administration of China, 2006.

[16] GB/T 32907-2016, *Information Security Technology — SM4 Block Cipher Algorithm*, Standardization Administration of China, 2016.

[17] International Standard, *ISO/IEC 18033-3:2010/AMD1:2021, Amendment 1 - Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers - SM4*, International Standardization of Organization (ISO), 2021.

[18] L. Goubin, J. M. Masereel, and M. Quisquater, "Cryptanalysis of white box DES implementations," in *International Conference on Selected Areas in Cryptography (SAC 2007)*, LNCS, vol. 4876, pp. 278–295, Springer, 2007.

[19] M. Jacob, D. Boneh, and E. Felten, "Attacking an obfuscated cipher by injecting faults," in *ACM Workshop on Digital Rights Management (DRM 2002)*, LNCS, vol. 2696, pp. 16–31, Springer, 2003.

[20] M. Karroumi, "Protecting white-box AES with dual ciphers," in *International Conference on Information Security and Cryptology (ICISC 2010)*, LNCS, vol. 6829, pp. 278–291, Springer, 2011.

[21] X. J. Lai, "Higher order derivatives and differential cryptanalysis," Communications and Cryptography: Two Sides of One Tapestry, no. 1994, pp. 227–233, Springer Science & Business Media, 1994.

[22] T. Lepoint, M. Rivain, Y. De Mulder, P. Roelse, and B. Preneel, "Two attacks on a white-box AES implementation," in *International Conference on Selected Areas in Cryptography (SAC 2013)*, LNCS, vol. 8282, pp. 265–285, Springer, 2014.

[23] T. T. Lin and X. J. Lai, "Efficient attack to white-box SMS4 implementation (in Chinese)," *Journal of Software*, vol. 24, no. 9, pp. 2238–2249, 2013.

[24] T. T. Lin, H. L. Yan, X. J. Lai, Y. X. Zhong, and Y. Jia, "Security evaluation and improvement of a white-box SMS4 implementation based on affine equivalence algorithm," *The Computer Journal*, vol. 61, no. 12, pp. 1783–1790, 2018.

[25] H. E. Link and W. D. Neumann, "Clarifying obfuscation: Improving the security of white-box DES," in *International Conference on Information Technology: Coding and Computing (ITCC 2005)*, vol. 1, pp. 679–684, IEEE, 2005.

[26] J. Q. Lu, J. Y. Li, "Cryptanalysis of two white-box implementations of the SM4 block cipher," in *International Conference on Information Security (ISC 2021)*, LNCS, vol. 13118. pp. 54–69, Springer, 2021.

[27] R. Luo, X. J. Lai and R. You, "A new attempt of white-box AES implementation," in *Proceedings 2014 IEEE International Conference on Security, Pattern Analysis, and Cybernetics (SPAC 2014)*, pp. 423-429, IEEE, 2014.

[28] W. Michiels, P. Gorissen, and H. D. L. Hollmann, "Cryptanalysis of a generic class of white-box implementations," in *International Conference on Selected Areas in Cryptography (SAC 2008)*, LNCS, vol. 5381, pp. 414–428, Springer, 2008.

[29] FIPS-197, *Advanced Encryption Standard (AES)*, National Institute of Standards and Technology (NIST) , 2001.

[30] FIPS-46, *Data Encryption Standard (DES)*, National Bureau of Standards (NBS), 1977.

[31] P. Shang, "White-box cryptography algorithm design and implementation of SMS4 (in Chinese)," Master's thesis, University of Electronic Science and Technology of China, 2016.

[32] Y. Shi, W. J. Wei, and Z. J. He, "A lightweight white-box symmetric encryption algorithm against node capture for WSNs," *Sensors*, vol. 15, no. 5, pp. 11928–11952, 2015.

[33] L. Tolhuizen, "Improved cryptanalysis of an AES implementation," in *Proceedings of the 33rd WIC Symposium on Information Theory in the Benelux and The 3rd Joint WIC / IEEE Symposium on Information Theory and Signal Processing in the Benelux*. pp. 68–71, Werkgemeenschap voor Informatie- en Communicatietheorie (WIC), 2012.

[34] R. S. Wang, H. Guo, J. Q. Lu, and J. W. Liu, "Cryptanalysis of a white-box SM4 implementation based on collision attack," *IET Information Security*, [Online]. Available: https://doi.org/10.1049/ise2.12045

[35] Z. Wu, J. Bai, D. S. Li, B. Li, B. Zeng, and Z. Q. Zhang, "White-box cryptographic video data sharing system based on SM4 algorithm (in Chinese)," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 46, no. 9, pp. 1660–1669, 2020.

[36] B. Wyseur, W. Michiels, P. Gorisseii, and B. Preneel, "Cryptanalysis of white-box DES implementations with arbitrary external encodings," in *International Conference on Selected Areas in Cryptography (SAC 2007)*, LNCS, vol. 4876, pp. 264–277, Springer, 2007.

[37] Y. Y. Xiao and X. J. Lai, "A secure implementation of White-Box AES," in *Proceedings of the Second International Conference on Computer Science and its Applications (CSA 2009)*, pp. 1–6, IEEE, 2009.

[38] Y. Y. Xiao and X. J. Lai, "White-box cryptography and a SMS4 implementation (in Chinese)," in *Proceedings of 2009 Annual Conference of the Chinese Association of Cryptologic Research*, pp. 24–34, 2009.

[39] S. Yao and J. Chen, "A new method for white-box implementation of SM4 algorithm (in Chinese)," *Journal of Cryptologic Research*, vol. 7, no. 3, pp. 358–374, 2020.