Analytic Toolbox for White-Box Implementations: Limitation and Perspectives

Chung Hun Baek, Jung Hee Cheon, and Hyunsook Hong

CHRI & Department of Mathematical Sciences, Seoul National University,
1 Gwanak-ro, Gwanak-gu, Seoul 151-747, Korea
{love0116,jhcheon,hongsuk07}@snu.ac.kr

Abstract. White-box cryptography is an obfuscation technique to protect the secret key in the software implementations even if an adversary has full access to the implementation of the encryption algorithm and full control over its execution platforms. This concept was presented in 2002 by Chow et al., and since then there have been many proposals to give solutions for the white-box cryptography. However, the progress does not seem to be substantial in spite of its practical importance. In fact, it is repeated that as a proposal on white-box implementation is announced, an attack of this implementation with lower complexity followed soon. It is mainly because most cryptanalytic methods were just targeted to some specific implementations and there is no general attack tool for the white-box cryptography.

In this paper, we present a general analytic toolbox for white-box implementations which extracts the secret information obfuscated in the implementation. For a general SLT cipher on n bits with S-boxes on m bits, one can remove the nonlinear encodings with complexity $O(\frac{n}{m_Q}2^{3m_Q})$ using our attack tool, if m_Q-bit nonlinear encodings are used to obfuscate input/output values in the implementation. Also, one can recover the affine encoding A in time $O(\frac{n}{m}\cdot {m_A}^3 2^{3m})$ using our extended affine equivalence algorithm (EAEA), if the inverse of the encoded round function F on n bits is given, where m_A is the smallest integer p such that A or its similar matrix obtained by permuting rows and columns is a block diagonal matrix with a $p \times p$ matrix as a block.

To avoid our attack, we need to consider a special encoding of large m_A, up to n. This results in storage blowing up in general. We suggest one approach with special affine encodings of $m_A = n$ that saves storage. In that case, the EAEA has the complexity $O\left(\min\left\{\frac{n}{m}\cdot n^{m+3}\cdot 2^{2m}, n\cdot\log n\cdot\sqrt{2}^n\right\}\right)$, which can be large up to 2^{74} and 2^{109} for $n = 128$ and 256, respectively, when $m = 8$. This gives an approach to design secure white-box implementation with practical storage. We expect that our analytic toolbox initiates the research on white-box implementation design.

Keywords: white-box cryptography, white-box implementation, extended affine equivalence algorithm, AES, block cipher.

1 Introduction

Traditionally, most of cryptographic algorithms are designed under the assumption that they are used in a trusted environment – trusted users, platforms, communication environments, etc. Under this assumption, cryptographic algorithms are designed to prevent attackers from getting the secret information using only input/output values of algorithms. In the real world, however, many untrusted hosts may try to access unapproved contents illegally, malicious softwares in the device of user can access memory used in execution of cryptographic algorithm, or much internal information can be leaked in communicating process. Hence, cryptographic algorithms should be designed with consideration for various and more powerful attacks. The concept of white-box cryptography has been proposed to satisfy these requirements.

The concept of white-box cryptography was presented in 2002 by Chow et al. The white-box cryptography is defined as an obfuscation technique which gives a secure software implementation, that even attackers who have full access to the implementation can not extract secret key information. In the past, the hardwares such as smart card, trusted platform module (TPM) were used to protect the internal information. Such hardware is costly and difficult to be replaced by a new one when a flaw is discovered. The white-box cryptography is a means to protect the internal information in the software implementation. The white-box cryptography can be used in many commercial products. One of the main applications is the digital right management (DRM). Suppose that some contents are in

encrypted form and it should be decrypted only in permitted devices. If an adversary obtains the decryption key of the contents, she can use them in other devices and distribute illegal copies of the contents. The white-box cryptography aims to prevent attackers obtaining the decryption key.

The first proposals to implement cryptographic primitives in white-box cryptography were done by Chow et al., who presented a white-box AES implementation [6] and a white-box DES implementation [7] in 2002. Both of them are broken in the sense that some attacks with lower complexity than their claimed security are announced: the attacks with complexity lower than 2^{14} for DES [19] and 2^{22} for AES [13]. After that, there have been many other approaches [5, 20, 12] to improve the white-box implementations using the concept of perturbations, wide linear encoding, and properties of dual cipher, respectively. However all of them have been also broken in lower complexity than their claimed security: with complexity 2^{17}, 2^{32}, and 2^{22}, respectively [16, 15, 13].

As we can see in the previous implementations, it is very difficult to design a white-box implementation of security level similar to the black-box model. Hence, the practical objective of the white-box implementations is to increase the complexity of cryptanalysis for the white-box implementation. However, all of the above implementations suffered strong attacks after their designs were announced. It is mainly because there is no standard attack tools such as differential cryptanalysis and linear cryptanalysis for block ciphers. Actually, all of the previous cryptanalysis for white-box implementations depends on ciphers or encodings. This can be a reason that there is no implementation with stable security, even though many approaches are attempted.

Our Contributions: In this paper, we develop general attack tools for the white-box implementations of SLT ciphers using table lookups. Let $E = M \circ S$ be the round function of a SLT cipher on n bits where M is an invertible linear map and S is a concatenation of S-boxes on m bits with a fixed key. We define the input encoding as $f = A \circ P$ where A is an invertible linear map and P is a concatenation of small nonlinear permutations. If we let g be the input encoding of the next round, then the encoded round function F of E is of the form $F = g^{-1} \circ E \circ f = QBSAP$ where B is an invertible linear map and Q is a concatenation of small nonlinear permutations.

Our toolbox consists of several algorithms to recover nonlinear and affine encodings used in this general model. First, by adopting the Biryukov-Shamir technique [4], we show that the nonlinear part Q can be removed up to affine transformation in $O\left(\frac{n}{m_Q}2^{3m_Q}\right)$, when $Q = (Q_1, \cdots, Q_{n/m_Q})$ and each Q_i is a permutation on m_Q bits. For example, the nonlinear encoding of Chow et al.'s can be removed in 2^{18} bit operations while it takes 2^{29} bit operations using Billet et al.'s attack [2]. While Billet et al.'s method is available only when the input size of the S-boxes is same to the input size of the encodings, ours can be efficiently applicable for the case where $m \neq m_Q$.

Second, when $F = B \circ S \circ A$ for affine mappings A, B, it is affine equivalent to S and hence we can apply the affine equivalence algorithm in [3], which result in $O(n^3 2^{2n})$ complexity. We improve this algorithm for the case where S consists of small S-boxes of size m. According to our extended affine equivalence algorithm (EAEA), we can find A and B in $O\left(\frac{n}{m} \cdot {m_A}^3 2^{3m}\right)$, where m_A is the input size of A. Using this attack, we can show that whatever output encoding is used the first round key can be found in $O(\frac{n}{m} \cdot m^3 \cdot 2^m)$ in this framework if the input encoding is not used.

Our attacks are universal in the sense that all known implementations are susceptible by our attacks. It gives negative perspective about secure white-box implementation of SLT ciphers with table lookups: The attack complexity is at most 2^{54} for $n \leq 256$ if the F^{-1} oracle is given. However, if we use an affine encoding A with $m_A = n$, it is hard to compute inverse of F, where m_A is the smallest integer p such that A or its similar matrix obtained by permuting rows and columns is a block diagonal matrix with a $p \times p$ matrix as a block. In this case, recovering the affine encodings by our EAEA takes $O\left(\min\left\{\frac{n}{m} \cdot n^{m+3} \cdot 2^{2m}, n \cdot \log n \cdot \sqrt{2^n}\right\}\right)$ which can be large enough up to 2^{74} and 2^{109} for $n = 128$ and 256, respectively. One shortcoming of this approach is large storage requirement.

However, it can be compensated by a use of special sparse encodings. We give an instance with the storage about 16 MB and 64 MB for one round when $n = 128, 256$, respectively, in the Section 4.

Outline of the Paper: In Section 2 we define a generalized white-box implementation model which would be a target of our general attack tools. We propose attack tools which can be applied to a white-box implementation in Section 3. An approach to the design of white-box implementation based on the result of our toolbox is given in Section 4. We conclude in Section 5.

2 White-Box Implementation Model

Throughout this paper, we focus on SLT ciphers on which most of previous works on white-box implementation are based. In this section, we first define SLT cipher and then present a generalized model of its white-box implementation using table lookups. Several instances of this generalized model will be found in Appendix A.

2.1 SLT Cipher

A substitution-linear transformation (SLT) cipher defined in [14] is a type of iterated cipher with a couple of substitution layers and linear transformation layers. SLT cipher can be considered ciphers using substitution-permutation network (SPN)[1], because permuting bits is a linear transformation. More precise definition of SLT cipher is as follows.

Definition 1. *A **SLT cipher** $\mathcal{E}$ is defined as follows. It consists r rounds for some $r \geq 1$. For each $i = 1, \cdots, r$, the i-th round function $E^{(i)}(x_1, \cdots, x_k)$ is a bijective function on n bits, where $n = m \cdot k$ and x_j is an m-bit value for each j and consists of following three operations.*

1. ***Xoring round key*** *Xor the i-th round key $K^{(i)} = (K_1^{(i)}, \cdots, K_k^{(i)})$ of n bits to the input $(x_1, \cdots, x_k)$. This outputs $y_j = x_j \oplus K_j^{(i)}$ for all $j = 1, \cdots, k$.*
2. ***Substitution*** *Compute $z_j = S_j^{(i)}(y_j)$ for all $j = 1, \cdots, k$, where each $S_j^{(i)}$ is an invertible S-box on m bits in the i-th round.*
3. ***Linear transformation*** *For $z = (z_1, \cdots, z_k)$, compute $M^{(i)}z$ where $M^{(i)}$ is an $n \times n$ invertible matrix over GF(2). This n-bit value is the output of the i-th round function.*

Note that operation 1,2 realize confusion and operation 3 realizes diffusion.

2.2 A Generalized White-Box Implementation Model

In the black-box model, it is assumed that the encryption algorithm is executed in trusted platforms. Hence, adversary can observe no internal behaviors of the encryption process but only external values, plaintext/ciphertext of encryption algorithm. However, these models are theoretical and the leakage of security information can occur in implementation process in practice. In the gray-box models, adversary can access more information about internal details of the encryption algorithm besides input/output values of the encryption. This information includes side channel information related to running time, power consumption, and fault analysis, which can be leaked by partial access.

On the other hand, in the white-box model, it is assumed that the adversary has full access to the implementation of the encryption algorithm and full control over its execution platforms. In this context, the main objective of the adversary is to extract the secret key. That is, the purpose of secure white-box implementations is to prevent the encryption key from being revealed even if internal

[1] A SPN is a type of iterated cipher with a couple of substitutions and permutation on bits.

algorithm details are completely visible in the untrusted platform and the adversary has full access to the execution of the encryption algorithm.

The naive approach to secure white-box implementation of an encryption is to give a table of all input/output values of the encryption. In this case, the security of the implementation of algorithm is equivalent to the security of the encryption in the black-box model and hence depends on the security of the encryption scheme itself, regardless of implementations. Unfortunately, such implementation is not practical because the storage of the table is quite large to implement. For example, the size of input/output table of AES-128 is $2^{128} \times 128 = 2^{102}GB$. A natural way to reduce the size of table is to decompose the table into "small" tables so that their composition is equivalent to the original input/output table.

Let $\mathcal{E} = E^{(r)} \circ \cdots \circ E^{(1)}$ be a SLT cipher defined in Definition 1. The most important factor of the table size is the number of input bits, and so the number of input bits which affect each S-box, since a linear function can be easily expressed in the summation of several linear functions with small inputs. Even if the S-box used in the block cipher has small input size and the number of input bits which affect each S-box is small in a single round,

each input bit affects all S-boxes by diffusion effect after several rounds. For example, AES-128 consists of 10 rounds and each round is made up of diffusion layer (MixColumn and ShiftRow) and confusion layer (SubBytes and AddRoundKey). In a single round, one S-box is influenced by only 8 input bits, but in more than two rounds, each S-box is influenced by all input bits. Therefore, a natural way of providing white-box implementation with tables of small size is to give input/output tables by round and decompose each input/output table of a round into small tables by input related to S-boxes so that their composition is equivalent to the input/output table of the round. Since the round key can be exposed if the input/output values of a single round are observed, we provide input/output tables of each round only after obfuscation by input/output encoding functions.

$$\underbrace{M_{out} \circ f^{(r+1)}}_{\text{table}} \circ \underbrace{(f^{(r+1)})^{-1} \circ E^{(r)} \circ f^{(r)}}_{\text{table}} \circ \cdots \circ \underbrace{(f^{(2)})^{-1} \circ E^{(1)} \circ f^{(1)}}_{\text{table}} \circ \underbrace{(f^{(1)})^{-1} \circ M_{in}}_{\text{table}}$$
$$= \quad M_{out} \circ E^{(r)} \circ \cdots \circ E^{(2)} \circ E^{(1)} \circ M_{in}$$

Fig. 1: The general strategy of white-box implementation

Let us consider a white-box implementation of $\mathcal{E}$ along the above strategy. Each round function $E^{(i)}$ is obfuscated by an input/output encoding function $f^{(i)}$ which is a bijection on n bits. The input encoding $f^{(i)}$ of i-th round is offset by the output encoding $(f^{(i)})^{-1}$ of the previous round. Then the i-th encoded round function is defined as $F^{(i)} = (f^{(i+1)})^{-1} \circ E^{(i)} \circ f^{(i)}$ for $i = 1, \cdots, r$. Additionally, external input/output encodings M_{in} and M_{out} are supplement for security. As a result, such white-box implementation is the same as the original encryption $\mathcal{E}$ except the effect of M_{in} and M_{out} on the input and output of $\mathcal{E}$.

Now, consider the input/output table for a round. We define $\oplus_c$ as the map $\oplus_c(x) = x \oplus c$, then we can write a round function $E^{(i)} = M^{(i)} \circ S^{(i)} \circ \oplus_{K^{(i)}}$, where $S^{(i)}$ is a concatenation of S-boxes $S_1^{(i)}, \cdots, S_k^{(i)}$ on m bits. We also define $f_j^{(i)} = \pi_j \circ f^{(i)}$ where π_j is a projection onto the j-th block corresponding to $S_j^{(i)}$. In addition, we let $M^{(i)} = \begin{bmatrix} M_1^{(i)} & \cdots & M_k^{(i)} \end{bmatrix}$ where $M_j^{(i)}$ is the j-th vertical strip of size $n \times m$. As mentioned earlier, we decompose the encoded input/output table of F_i into several tables with small inputs. For each $1 \leq j \leq k$, the j-th table is the input/output table of

$$F_j^{(i)} = (f^{(i+1)})^{-1} \circ M_j^{(i)} \circ S_j^{(i)} \circ \oplus_{K_j^{(i)}} \circ f_j^{(i)}$$

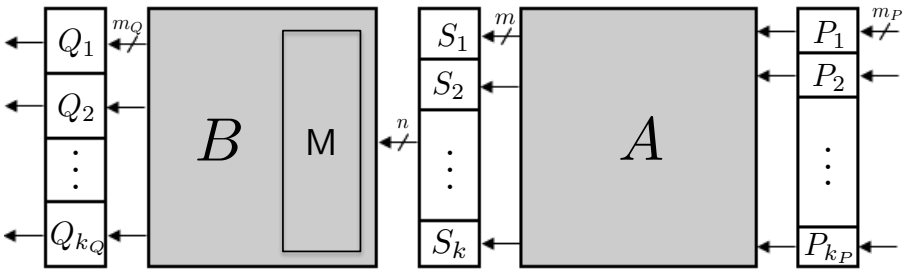

Fig. 2: The general form of the encoded function in white-box implementations

and hence, the input size of this table is determined by that of $f_j^{(i)}$. Furthermore, we can obtain the output of $F^{(i)}$ by the following calculation

$$F^{(i)}(x) = \left(f^{(i+1)}\right)^{-1} \circ \left(\sum_{j=1}^{k} f^{(i+1)} \circ F_j^{(i)}(x)\right).$$

The last calculation is performed directly only when $f^{(i+1)}$ is linear. If $f^{(i+1)}$ has nonlinear factor, we need additional tables, called XOR tables, to perform the last calculation as in [6].

Therefore, the form of white-box implementation is determined by encoding $f^{(i)}$. For example, in Chow et al.'s white-box AES implementation [6], $f^{(i)}$ is a block diagonal mapping with block size 8 and the encoding on each block is composed of linear mixing bijection and nonlinear permutation. So, the $f_j^{(i)}$ is considered as an 8-bit encoding with net input size 8. Furthermore, since the encodings have nonlinear factors, XOR tables are required.[2]

Now we set a generalized white-box implementation model, which would be a target of our attack tool, using more general encoding function $f^{(i)}$. After here, we omit the index for round i, if there is no confusion, and let $f = f^{(i)}$ and $g = (f^{(i+1)})^{-1}$. Our candidate of the general encoding is $f = A \circ P$ where A is an invertible linear map on n bits and P is a nonlinear permutation. However, if A and P are arbitrary bijective linear and nonlinear mappings, resp., the table size would be huge. So, we consider a special form of mappings that gives a white-box implementation with small table size.

Let $A = \begin{bmatrix} A_1 \\ \vdots \\ A_k \end{bmatrix}$ where A_j is the j-th horizontal strip of size $m \times n$. Then $f_j = A_j \circ P$ and hence the input size of f_j is related to the input size of A_j. Since the input size of A_j is the number of nonzero columns in A_j, to get a small table the number of nonzero columns in A_j should be small. Furthermore, since these net input bits are affected by corresponding P_{j_t}'s, the number of such P_{j_t} have to be few and each of P_{j_t}'s must have small number of input bits. Therefore, P should be a concatenation of small nonlinear permutations and A should be an invertible linear map, where each A_j has small number of nonzero columns.

Since the output encoding is the inverse of the input encoding of the next round, output encoding g is of the form $g = Q \circ B$ where Q is a concatenation of small nonlinear permutations and B is an invertible linear map. Thus the encoded round function is of the form $F = QBMSAP$.[3] Here, we omit M because B, M are all invertible linear maps and M is known. Putting all the above discussion together, we define a generalized model of white-box implementation for the SLT cipher using table lookups as follow.

[2] This XOR tables can be decomposed into 4-bit blocks because the nonlinear encodings are block diagonal mappings with block size 4.

[3] For a fixed key, the adding key operation can be merged to the nonlinear permutation.

Definition 2. *Let $E = M \circ S$ be a round function of SLT cipher with block size n, where M is a linear mapping on n bits and S is a layer of S-boxes on m bits. Then a generalized form of the encoded round function F of E in the white-box implementation is defined by $F = QBSAP$, where P, Q are layers of small nonlinear permutations, A, B are layers of linear mapping on n bits, and each A_j has small number of nonzero columns (A_j is the j-th $m \times n$ horizontal strip of A).*

We may consider the case of $f = P \circ A$, where the encoded round function is of the form $F = BQMSPA$. In this case, since the $n \times n$ linear map B follows the Q layer, we cannot add up the outputs of each table using XOR tables. Therefore we have to decompose F into two parts after Q layer. That is, we let $F = G \circ H$ and make input/output tables of G and H, where $G = B \circ Q_1$ and $H = Q_2MSPA$ with $Q = Q_1 \circ Q_2$. However, if we combine H with G of the previous round, the function is of the form $Q'MSP'$ because the linear mappings A and B (of previous round) will be canceled out. Since this case is included in the case of $f = A \circ P$, we do not treat it any more. It is similar in the case that a composition of more than 2 encodings is used. Therefore we just consider the case of $f = A \circ P$ as a generalized form of encoding.

3 General Attack Toolbox for White-Box Implementation

In white-box cryptography, the attacker's objective is to extract the secret key information. Most block ciphers have key schedules, and so the cryptanalysis focuses on recovering one round key. In order to extract the secret key, we find the encodings of consecutive two round functions. Using the relation between the output encodings and the input encodings of the consecutive two rounds, the secret key can be extracted efficiently. Therefore, the goal of this section is the extraction of the secret encodings obfuscated in implementation.

On the other hand, in previous white-box implementations, cryptanalysis is specialized for the target implementation. We introduce general tools to recover encodings in $F = QBSAP$, general model of white-box implementations we defined in the previous section. We first recover nonlinear parts of encodings up to affine transforms and then we can let $F = B \circ S \circ A$, where A, B are invertible affine maps. Next, we propose attack tools to find A and B in various cases.

3.1 Recovering Nonlinear Encodings

Throughout this subsection, we let $F = QBSAP$ be a round function of white-box implementations with input size n, where S is a layer of k S-boxes on m bits, A, B are layers of invertible affine mapping on n bits, and P, Q are layers of k_P nonlinear bijective encodings on m_P bits and k_Q nonlinear bijective encodings on m_Q bits, respectively. Furthermore, if A is a block diagonal map consisting in mixing bijections on each block, we also let m_A be the size of blocks and k_A the number of blocks (*i.e.*, $n = k \cdot m = k_P \cdot m_P = k_Q \cdot m_Q = k_A \cdot m_A$).

Usually, recovering nonlinear parts of encoding is very difficult, but in white-box implementations it is easier because only small nonlinear encodings are used. Billet et al. [2] presented a method to recover nonlinear parts of encodings in Chow et al.'s implementation [6] in 2^{3m} steps. Billet et al. applied this method to only the case that the size of encoding blocks is the same as the size of S-boxes, more precisely, $\mathrm{lcm}(m_P, m_A, m_Q) = m$ where lcm means the least common divisor. Actually, in Chow et al.'s implementation [6] the size of the S-boxes and the mixing bijections is 8 and the size of the nonlinear encodings is 4. This method easily can be extended to the case that $\mathrm{lcm}(m_P, m_A, m_Q)$ divides m with same complexity by regarding $\frac{m}{m_P}$ encodings in layer P, $\frac{m}{m_A}$ mixing blocks in layer A and $\frac{m}{m_Q}$ encodings in layer Q as a single encoding in the P, A, Q layers, respectively.

How about the case that $\mathrm{lcm}(m_P, m_A, m_Q)$ does not divide m? In this case, the method also can be applied to the implementation if $\mathrm{lcm}(m_P, m_A, m_Q, m) < n$, by considering $\mathrm{lcm}(m_P, m_A, m_Q, m)$ as the size of encodings in the P, A, Q and S layers. However, the complexity of this attack is

$2^{3\mathrm{lcm}(m_P,m_A,m_Q,m)}$, and no longer depend only m. For example, consider the case that $n = 192, m_P = m_Q = 6$ and $m_A = m = 8$, the method has complexity 2^{74}. This gives the following theorem which is extended version of Billet et al.'s attack.

Theorem 1. *(Let $F = QBSAP$ be a round function of white-box implementations defined above. If $l = lcm(m_P, m_A, m_Q, m) < n$, then one can recover a nonlinear part Q (up to affine transformation) in time $\frac{n}{l} \cdot 2^{3l}$.*

In this subsection, we introduce a more efficient tool to recover nonlinear parts of encodings for the latter case, which is based on the multiset attack of Biryukov and Shamir [4]. Using this tool, we can recover nonlinear parts of encodings efficiently even if the size of linear mixing bijections is larger than the size of the S-boxes or the size of nonlinear encodings is different from the size of the S-boxes.

In order to explain this tool, we will use the multiset properties as in [4]. For more general attack, we added a subscript to each property symbol to denote the size of input. For a multiset M of m-bit values ($m > 1$), the multiset properties are defined as follows:

- M has property C_m (constant) if it contains only numbers of a single m-bit value.
- M has property P_m (permutation) if it contains all numbers of the 2^m possible values exactly once.
- M has property E_m (even) if each value occurs an even number of times or does not occur.
- M has property B_m (balanced) if the XOR of all the values is 0^m.

We extend this notation to denote combined properties. First, we define a projection map $\pi_I : \{0,1\}^n \to \{0,1\}^\tau$ by $\pi_I(x_1, \cdots, x_n) = (x_{i_1}, \cdots, x_{i_\tau})$, for index set $I = \{i_1, \cdots, i_\tau\} \subseteq \{1, \cdots, n\}$. We say a multiset M of n-bit values has property $\mathsf{P}_{2m}^k\mathsf{C}_{n-2km}$, if $\pi_{\{2im+1,\cdots,2im+2m\}}(M)$ has property P_{2m} for each $i = 0, \cdots, k-1$ and $\pi_{\{2km+1,\cdots,n\}}(M)$ has property C_{n-2km}.

Now let us consider how the multiset properties are transformed by an affine mapping, in the following two lemmas. See Appendix B for the proofs.

Lemma 1. *Let $A : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ be an affine mapping and $I = \{i_1, \cdots, i_\tau\} \subseteq \{1, \cdots, n\}$ with $\tau \geq m > 1$. For a multiset M of n-bit values, a multiset $A(M)$ has property P_m or E_m if $\pi_I(M)$ has property P_τ, $\pi_{\{1,\cdots,n\}\setminus I}(M)$ has property $\mathsf{C}_{n-\tau}$.*

Lemma 2. *Let $A : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ be an affine mapping. For a multiset M of n-bit values, the multiset $A(M)$ of m-bit values has property B_m if M has property B_n and the size of M is even.*

Using these lemmas, we obtain the following theorem, a generalized version of the result in [4]. This attack tool which can remove the nonlinearity of encodings is more efficient than Billet et al.'s attack.

For description of the theorem, we provide some definitions. We say a function $f : \mathbb{Z}_2^u \to \mathbb{Z}_2^v$ with $u \geq v$ is balanced if every output occurs 2^{u-v} times. We define $F^{i,\alpha} : \mathbb{Z}_2^{i \cdot m_P} \to \mathbb{Z}_2^n$ as $F^{i,\alpha}(x) := F(\alpha_1, x, \alpha_2)$ where $\alpha = (\alpha_1, \alpha_2)$ and $\alpha_1 \in \mathbb{Z}_2^{t \cdot m_P}, \alpha_2 \in \mathbb{Z}_2^{n-(i+t)\cdot m_P}$ for some $0 \leq t \leq k_P - i$ and $F_j^{i,\alpha} := \pi_j \circ F^{i,\alpha}$, where π_j is a projection onto the j-th block of layer Q. Lastly, we define a set of functions $\Lambda_{i,j} = \{F_j^{i,\alpha} \mid \alpha \in \mathbb{Z}_2^{t \cdot m_P} \times \mathbb{Z}_2^{n-(i+t)\cdot m_P} \text{for some } 0 \leq t \leq k_P - i\}$.

Theorem 2. *Let $F = QBSAP$ be a round function of white-box implementations and $\Lambda_{i,j}$ a set of functions defined above where $i = \lceil \frac{m}{m_P} \rceil$. Assume the probability that a function in $\Lambda_{i,j}$ is not balanced is at least $\mathfrak{p} > 0$ for each j. If $lcm(m_P, m_A, m_Q)$ does not divide m, then one can recover nonlinear part Q (up to affine transformation) using $2^{i \cdot m_P + m_Q} \cdot O(1/\mathfrak{p})$ chosen plaintexts in about $O(k_Q \cdot 2^{3m_Q})$ bit operations.*

Proof. Let α be an $(n - i \cdot m_P)$-bit value. For some t, take M_α to be a set with property $\mathsf{C}_{m_P}^t\mathsf{P}_{(i\cdot m_P)}\mathsf{C}_{m_P}^{k_P-(i+t)}$ such that $\pi_{\{1,\cdots,t\cdot m_P,(i+t)m_P+1,\cdots,n\}}(x) = \alpha$ for each $x \in M_\alpha$.

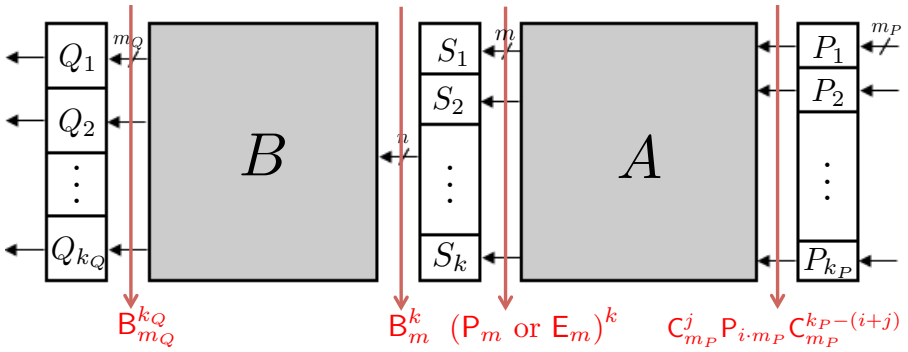

Fig. 3: The relations between multiset properties on QBSAP

The property $\mathsf{C}^t_{m_P}\mathsf{P}_{(i\cdot m_P)}\mathsf{C}^{k_P-(i+t)}_{m_P}$ is preserved by the layer P, and thus output multiset has also property $\mathsf{C}^t_{m_P}\mathsf{P}_{(i\cdot m_P)}\mathsf{C}^{k_P-(i+t)}_{m_P}$. Since A can be divided into k affine mappings from n-bit to m-bit and $i \cdot m_P \geq m$, this property is transformed by the layer A into the multiset with property $(\mathsf{P}_m$ or $\mathsf{E}_m)^k$ by Lemma 1. Since the Property $(\mathsf{P}_m$ or $\mathsf{E}_m)^k$ is preserved after layer S, the multiset after layer S has the property B^k_m and this property is equivalent to property B_n. By Lemma 2, the property B_n is transformed by the layer B into the multiset with property $\mathsf{B}^{k_Q}_{m_Q}$ by dividing B into k_Q affine mappings from n-bit to m_Q-bit.

Now, consider the j-th nonlinear bijective encoding Q_j in the layer Q and define $F_j = \pi_j \circ F$, where π_j is a projection onto the j-th block. Then we get a homogeneous equation

$$\sum_{x \in M_\alpha} Q_j^{-1}(F_j(x)) = 0^{m_Q}$$

and since we know the values of $F_j(x)$ for all $x \in M_\alpha$, this equation is a homogeneous equation of the unknowns $Q_j^{-1}(i)$'s for all i through m_Q-bit values, *i.e.*

$$\sum_i c_{\alpha,i} Q_j^{-1}(i) = 0^{m_Q}$$

where $c_{\alpha,i}$ is the number of $x \in M_\alpha$ satisfying $F_j(x) = i$.

Since the number of unknowns is 2^{m_Q}, we need more than 2^{m_Q} equations. If we use different constant α at the part correspond to property C from $\mathsf{C}^t_{m_P}\mathsf{P}_{(i\cdot m_P)}\mathsf{C}^{k_P-(i+t)}_{m_P}$, we are likely to get a different homogeneous equation of $Q_j^{-1}(i)$'s. By the assumption, we can obtain 2^{m_Q} equations from $2^{m_Q} \cdot O(1/\mathfrak{p})$ multisets, then we can solve the system of equations by Gaussian elimination. We can do this process for all j's and hence we need $O(k_Q 2^{3m_Q})$ bit operations with $2^{i\cdot m_P + m_Q} \cdot O(1/\mathfrak{p})$ chosen plaintexts to recover the layer Q up to affine transformation. □

To remove the nonlinearity of encodings, Billet et al.'s attack take $2^{3\mathrm{lcm}(m_P, m_A, m_Q, m)}$ bit operations, but our attack tool only takes 2^{3m_Q} bit operations. Reconsider example for $n = 192, m_P = m_Q = 6$ and $m_A = m = 8$. In this case, our attack tool reduces the complexity from 2^{74} to 2^{22} to remove the layer Q up to affine transformation.

Remark 1. To apply the method in Theorem 2, we require sufficiently many homogeneous equations of the form $\sum_{x \in M_\alpha} Q_j^{-1}(F_j(x)) = 0^{m_Q}$. It is related to the probability $\mathfrak{p}$ because if $F_j^{i,\alpha}$ is balanced, the equation is a trivial equation. By the nonlinearity of S-boxes, if $F_j^{i,\alpha}$ is related to more than 2 S-boxes, $F_j^{i,\alpha}$ is likely to be not balanced. So, we have to take care of choosing multiset of plaintexts, so that $F_j^{i,\alpha}$ is related to more than 2 S-boxes and we note that if $lcm(m_P, m_A, m_Q)$ does not divide m, $F_j^{i,\alpha}$

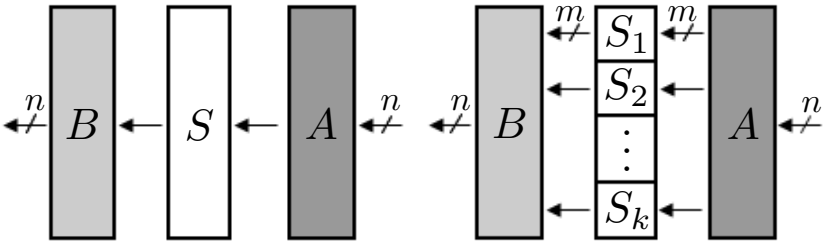

Fig. 4: Affine Equivalence problem and Extended Affine Equivalence problem

is related to more than 2 S-boxes. However, in the case that m_A, m_P and m_Q are equal to m, we can acquire only trivial equation, and hence we cannot use this method. Nevertheless, we can also recover the nonlinear parts of the encodings because Billet et al.'s method can be applied to this case (Billet et al.'s method has same complexity as the method in Theorem 2). Therefore, the toolbox to recover the nonlinearity of the encodings should include both methods with same complexities, considering all the cases.

Actually, we cannot recover Q exactly because we cannot get a system of equation with full rank of 2^{m_Q}, but we can recover Q up to affine transform. Furthermore, we can recover P by attack for the previous round. Therefore, if we assume $k_P = k_Q$, we can recover all nonlinear part of encoding of a round function in $2k_Q \cdot 2^{3m_Q}$ steps.

Applications In Chow et al.'s implementation [6], the input bit size of the linear encodings and the S-boxes is 8 and the size of input/output nonlinear encodings is 4: In our notations, $m = m_A = 8$ and $m_P = m_Q = 4$. Thus, applying the result of Theorem 2, we can recover the nonlinear encodings in $2k_Q \cdot 2^{3m_Q} = 2 \cdot 32 \cdot 2^{3 \cdot 4} = 2^{18}$ and the complexity is much less than Billet et al.'s [2], 2^{29}.

3.2 Extended Affine Equivalence Algorithm with Multiple S-boxes

We say that F and S are *linear/affine equivalent* if there exist linear/affine mappings A, B such that $F = B \circ S \circ A$. *The linear/affine equivalence problem* is to find invertible linear/affine mappings A and B for given nonlinear bijections S and $F = B \circ S \circ A$.

Biryukov et al. [3] proposed algorithm for solving the linear equivalence problem for arbitrary permutations over $\mathbb{Z}_2^n$ with complexity $O(n^3 2^n)$. For the affine equivalence algorithm, they proposed the concept of the representatives for the linear equivalence classes of permutations. Using the representatives, the affine equivalence problem can be solved in time $O(n^3 2^{2n})$.

In this subsection, we consider the case that the nonlinear mapping S consists of k invertible S-boxes S_i which maps from $\mathbb{Z}_2^m$ to $\mathbb{Z}_2^m$, where $n = km$ as Fig. 4. The problem may be considered to be an extension of [3] and so called *the Extended Affine Equivalence Problem*. The following theorem says the problem can be solved more efficiently when compared with the affine equivalence problem.

Theorem 3. *Let F and S be two permutations over $\mathbb{Z}_2^n$ and $S = (S_1, \cdots, S_k)$ with nonlinear permutations S_i over $\mathbb{Z}_2^m$ for $i = 1, \cdots, k$. Assume that we can easily access the inversion of F. Then, we can find all affine mappings A and B such that $F = B \circ S \circ A$ in time $O(kn^3 2^{3m})$ if they exist.*

Proof. First, we assume that F and S are linear equivalent. Suppose that A and B are invetible linear mappings over $\mathbb{Z}_2^n$ with $F = B \circ S \circ A$. Let us consider A and B^{-1} to be partitioned into k horizontal strips of size $m \times n$. Denote the i-th strip of A and B^{-1} by A_i and B_i respectively. That is,

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_k \end{bmatrix} \quad \text{and} \quad B^{-1} = \begin{bmatrix} B_1 \\ \vdots \\ B_k \end{bmatrix}. \tag{1}$$

If one can obtain two sets $\{x_1, x_2, \cdots, x_n\}$ and $\{B_i \circ F(x_1), B_i \circ F(x_2), \cdots, B_i \circ F(x_n)\}$ such that $\{F(x_1), F(x_2), \cdots, F(x_n)\}$ is linearly independent, then one can find B_i from

$$B_i = \left[B_i \circ F(x_1)\ B_i \circ F(x_2)\ \cdots\ B_i \circ F(x_n)\right]\left[F(x_1)\ F(x_2)\ \cdots\ F(x_n)\right]^{-1}, \tag{2}$$

where we consider $B_i \circ F(x_j)$ and $F(x_j)$ as column vectors for $1 \le j \le n$. Hence, the main strategy is to find two sets $\{x_1, \cdots, x_n\}$ and $\{B_i \circ F(x_1), \cdots, B_i \circ F(x_n)\}$ such that $\{F(x_1), \cdots, F(x_n)\}$ is linearly independent in order to recover B_i.

Suppose that we have two sets $\{x_1, \cdots, x_\ell\}$ and $\{y_1 = B_i \circ F(x_1), \cdots, y_\ell = B_i \circ F(x_\ell)\}$ such that $\{x_1, \cdots, x_\ell\}$ is linearly independent. For any $x = \sum_{j=1}^{\ell} b_j x_j$ ($b_j \in \{0, 1\}$), we can compute $y = B_i \circ F(x)$ from $y_1, \cdots, y_\ell$ by

$$y = S_i \circ A_i(x) = S_i\left(\sum_{j=1}^{\ell} b_j A_i(x_j)\right) = S_i\left(\sum_{j=1}^{\ell} b_j S_i^{-1}(y_j)\right). \tag{3}$$

Since F is a nonlinear bijection, we can obtain another vector x such that $F(x) \notin \mathbb{Z}_2 F(x_1) + \cdots + \mathbb{Z}_\ell F(x_\ell)$ with high probability. (Assuming F is random bijection, at least one of $\{F(x) \mid x \in \mathbb{Z}_2 x_1 + \cdots + \mathbb{Z}_2 x_\ell\}$ does not belong to $\mathbb{Z}_2 F(x_1) + \cdots + \mathbb{Z}_2 F(x_\ell)$ with probability $1 - \left(\frac{2^{d_\ell}}{2^n}\right)^{2^\ell - \ell}$ where $d_\ell = \dim\langle\{F(x_1), \cdots, F(x_\ell)\}\rangle$.)

On the other hand, suppose that we have two sets $\{F(x_1), \cdots, F(x_\ell)\}$ and $\{y_1 = B_i \circ F(x_1), \cdots, y_\ell = B_i \circ F(x_\ell)\}$ such that $\{F(x_1), \cdots, F(x_\ell)\}$ is linearly independent. For any $x' = F^{-1}(\sum_{j=1}^{\ell} b'_j F(x_j))$, we can compute $y' = B_i \circ F(x')$ from $y_1, \cdots, y_\ell$ by

$$y' = B_i \circ F\left(F^{-1}\left(\sum_{j=1}^{\ell} b'_j F(x_j)\right)\right) = \sum_{j=1}^{\ell} b'_j B_i \circ F(x_j) = \sum_{j=1}^{\ell} b'_j y_j. \tag{4}$$

Since F^{-1} is a nonlinear bijection then we can obtain a new vector x' such that $x' \notin \mathbb{Z}_2 x_1 + \cdots + \mathbb{Z}_2 x_\ell$ with high probability by assuming F^{-1} is random bijection.

Set $x_0 = 0$, $y_0 = B_i \circ F(x_0)$, $x_1 = F^{-1}(0)$ with $F(x_1) = 0$. Then we have $y_0 = S_i \circ A(x_0) = S_i(0), y_1 := B_i \circ F(x_1) = 0$. We need to make an initial guess $y_2 := B_i \circ F(x_2)$ for some $x_2 \in \{0,1\}^n \backslash \{x_0, x_1\}$ to generate another vectors. Note that x_1, x_2 are linearly independent. If we set $x_3 = x_2 + x_1$, then $F(x_3)$ does not belong to $\mathbb{Z}_2 F(0) + \mathbb{Z}_2 F(x_2)$ because F is nonlinear and $x_3 \notin \{x_0, x_1, x_2\}$. By repeating above process several times, we can successfully obtain n vectors whose F value are linearly independent. For each successful guessing, we get an $m \times n$ linear mapping B_i. If a mapping $S_i^{-1} \circ B_i \circ F$ is non-linear, we reject incorrect guesses. This process requires n^3 operations for each guessing, and thus the complexity becomes $kn^3 2^m$ to find full matrix B.

Now, let us consider the affine equivalence problem. An affine case is very similar to the linear case. Since an affine mapping is the composition of a linear map and a translation, we can write

$$B_i \circ F(x) + b_i = S_i\left(A_i(x) + a_i\right),$$

for $m \times n$ linear mappings A_i, B_i and the m-bit constant vectors a_i, b_i for $i = 1, \cdots, k$.

For each pair $(a_i, b_i) \in \mathbb{Z}_2^m \times \mathbb{Z}_2^m$, we follow the above process with inputs $F(x)$ and $S_i(x + a_i) + b_i$ and then we can solve the affine equivalence problem. Therefore, the total complexity is $O(kn^3 2^{3m})$ by additionally choosing two m-bit constant vectors. □

We call the algorithm in the Theorem 3 the *extended affine equivalence algorithm* (EAEA). While the affine equivalence algorithm has the complexity $O(n^3 2^{2n})$ to find the affine mappings A, B, the EAEA has only complexity $O(kn^3 2^{3m})$. This algorithm gives that the dominant parts of the complexities depend on m, not on n even though A and B are random affine mapping over $\mathbb{Z}_2^n$. Therefore, the EAEA is more efficient whenever S is a concatenation of several S-boxes.

Remark 2. The EAEA requires several evaluation of F^{-1} in equation (4) and so does not work when one can not access the inversion of F. Therefore, we have to consider the complexity for computing the inversion of F in order to use the EAEA.

When A is split We may consider $A \in (\mathbb{Z}_2)^{n\times n}$ as a $\tilde{A} \in (\mathbb{Z}_2^{m\times m})^{k\times k}$ with $n = km$. If $\tilde{A}$ is of form $\left[\begin{array}{c|c|c} * & 0 & * \\ \hline 0 & A^* & 0 \\ \hline * & 0 & * \end{array}\right]$ for $A^* \in (\mathbb{Z}_2^{m\times m})^{k_0\times k_0}$ and $k_0 \geq 1$, we say that A is *split*, and *unsplit* otherwise. If A is split, we can recover the encoding that corresponds to A^* with complexity $k_0(k_0m)^3 2^{3m}$. For more detail, refer to the appendix C.

Without the external encodings Suppose a white-box implementation of SLT cipher in our general model has no external encoding. Then we can recover the first round key in $O\left(\frac{n}{m} \cdot m^3 2^m\right)$: Because the encoded first round function F can be written as

$$B_i \circ F(0, \cdots, 0, x, 0, \cdots, 0) + B_i \circ F(0, \cdots, 0) = S_i(x + K_i) + S_i(K_i)$$

for all m-bit x, where $K = (K_1, \cdots, K_k)$ is the first round key on n bits with $n = km$, we need to guess only one m-bit K_i by taking A to the identity matrix. In this sense, the external encodings are essential and the encryption function E_K with a fixed key K should be replaced by the composition $M_{out} \circ E_K \circ M_{in}$, where M_{in} and M_{out} are external input/output encodings.

Applications In Xiao and Lai's implementation [20], they use only the linear mappings for input/output encoding. The input bit size of the input encodings is twice of the input bit size of the S-boxes. By fixing input value on all but 2 bytes as a constant, one can obtain the bijection map F on $\mathbb{Z}_2^{16}$ of the following form:

$$F = B \circ (S, S) \circ (\oplus_{K'}, \oplus_{K''}) \circ A$$

where A, B are linear invertible maps on 16 bits. Then F is affine equivalent to (S, S) with linear map B and affine map $(\oplus_{k'}, \oplus_{k''}) \circ A$.

By applying the extended affine equivalence algorithm, we can recover one part of the secret encoding in $\frac{n}{m} n^3 2^{2m} = 2^{29}$ steps for $m = 8$ and $n = 16$. This result is coincident with the result of Mulder et al. [15], $2n^3 2^n = 2^{29}$ steps. However, our attack tool has some potential advantages over Mulder et al.'s: (1) First, as n is larger than twice of m, *i.e.* $n = km$ with $k > 2$, our attack has less complexity than Mulder et al.'s. For the case of $m = 8$ and $n = 4m = 32$, the complexity to recover one part of the secret encoding is $\frac{n}{m} n^3 2^{2m} = 2^{33}$ using our attack tool, while $2n^3 2^n = 2^{48}$ using Mulder et al.'s method. (2) One additional advantage of our attack is that if we set A and B to be affine mappings instead of linear mappings to increase security, our tool can be applicable to the scheme while Mulder et al.'s method cannot. For the affine case with same n and m, one can recover a secret encoding in $\frac{n}{m} n^3 2^{3m} = 2^{37}$ using our tool.

4 Designing of White-Box Implementation

There have been many proposals for a new white-box implementation, but none of them appears to have more than 2^{32} complexity to recover the whole secret key. The urgent subject of white-box cryptography is to design the white-box implementation of higher security with the reasonable storage. In this section, we explore why the previous white-box implementations can be attacked with low complexity and how to design a white-box implementation to overcome this barrier. Note that we consider a SLT-type block cipher of n-bit inputs with m-bit S-boxes.

Recall that m_A is the size of minimized blocks of block diagonal affine encodings. More precisely, consider the affine encoding of the form $\oplus_a \circ A$ for A is an invertible matrix in $R^{k\times k}$ and a is a n-bit

value, where $R = \mathbb{Z}_2^{m\times m}$. Let k_0 be the smallest integer such that there exists two permutation matrices P_1 and $P_2 \in \mathbb{Z}_2^{km\times km}$ satisfying $P_1AP_2 = \left[\begin{array}{c|c} A_1 & \mathbf{0} \\ \hline \mathbf{0} & A_2 \end{array}\right]$ for some $A_1 \in R^{k_0\times k_0}$. We define $m_A = k_0 m$.

4.1 Limitation of White-Box Implementation: $m_A \leq 32$

Putting the above theorems together, we obtain the following theorem.

Theorem 4. (main theorem) *For $i = 1, 2, 3$, let $F_i = Q_iB_iS_iA_iP_i$, bijections on n bits and S_i, a concatenation of $\frac{n}{m}$ nonlinear bijection on m bits are given where P_i and Q_i are concatenations of $\frac{n}{m_Q}$ nonlinear bijection on m_Q bits, B_i is an invertible linear mapping on n bits and A_i is an invertible affine mapping on n bits with constant part K_i, satisfying $Q_i^{-1} = P_{i+1}$ and $A_{i+1} \circ B_i = \oplus_{K_{i+1}} \circ I_n$.*

Then one can find K_2 in time

$$O\left(3\frac{n}{max(m_Q, m)} \cdot 2^{3max(m_Q,m)} + 2\frac{n}{m} \cdot lcm(m_A, m_Q)^3 2^{3m}\right)$$

with $O\left(\frac{2n\log(lcm(m_A,m_Q))}{lcm(m_A,m_Q)}\right)$ calls of F_i^{-1} oracle, or in time

$$O\left(3\frac{n}{max(m_Q, m)} \cdot 2^{3max(m_Q,m)} + 2\tfrac{n}{m} \cdot lcm(m_A, m_Q)^{m+3} 2^{2m}\right)$$

without using F_i^{-1} oracle.

Proof. Note that $m|m_A$ by definition of m_A. Since $lcm(m_A, m_Q)|m$ implies $l = lcm(m_A, m_Q, m) = m$, one can recover Q_i (up to affine transformation) in time $O(\frac{n}{max(m_Q,m)} \cdot 2^{3max(m_Q,m)})$ by Theorem 1 and Theorem 2 and also can recover P_1 and P_2 from $P_1 = Q_0^{-1}$ and $P_2 = Q_1^{-1}$.

Now, for $i = 1, 2$, the nonlinear effects of P_i and Q_i can be removed in F_i and hence F_i can be considered $F_i = \tilde{B}_i \circ S_i \circ \tilde{A}_i$ where $\tilde{A}_i$ can be considered block diagonal affine mappings with block size $l = lcm(m_A, m_Q)$. Therefore, one can apply EAEA to each block of size l. When EAEA is applied to block of size l, it needs $\log l$ calls of F_i^{-1} oracle or to guess about $\log m_A$ vectors, instead of one vector, without using F_i^{-1} oracle. It follows that one can recover $\tilde{A}_i$ and $\tilde{B}_i$ in time $O(\frac{n}{m} \cdot l^3 2^{3m})$ with $\frac{n}{l}\log l$ calls of F_i^{-1} oracle or $O(\frac{n}{m} \cdot l^3 2^{m(\log l+2)}) = O\left(\frac{n}{m} \cdot l^{m+3} 2^{2m}\right)$ without using F_i^{-1} oracle.

From the relation between P_2 and Q_1, one can find $\tilde{A}_2 \circ \tilde{B}_1(0) = A_2 \circ B_1(0) = K_2$. □

All of the previous white-box implementations have common features: For $n = 128, m = 8$, (1) they use affine/linear encodings with $m_A \leq 16$ and (2) they don't use the nonlinear encodings or use the nonlinear encodings with only $m_Q = 4, 8$. In these case, $lcm(m_A, m_Q) \leq 16$ and so one can easily compute the inverse. By the result of Theorem 4, all of the previous can be broken in time less than 2^{41} without using the specific attacks. Actually, white-box implementations with $n = 128, m = 8, m_A \leq 32$ and $m_Q = 4$ or 8 can be broken in time at most 2^{44}.

To increase the complexity, we need to increase $lcm(m_A, m_Q)$. Increasing m_Q results in large storage due to XOR table, for example, the required storage of "one" XOR table is 8GB for $m_Q = 16$. Another approach is to increase m_A. It also increases the storage size in general, but it can be bounded for some special affine transformations. We explore its possibility in the next subsection.

4.2 Perspective of White-Box Implementation

We may try to increase m_A to get a higher attack complexity. Unfortunately, the complexity upper bounded by 2^{54} even for the largest $m_A = n$ when $n \leq 256$ when $m = 8$ if the F^{-1} oracle is given, where F is an encoded round function.

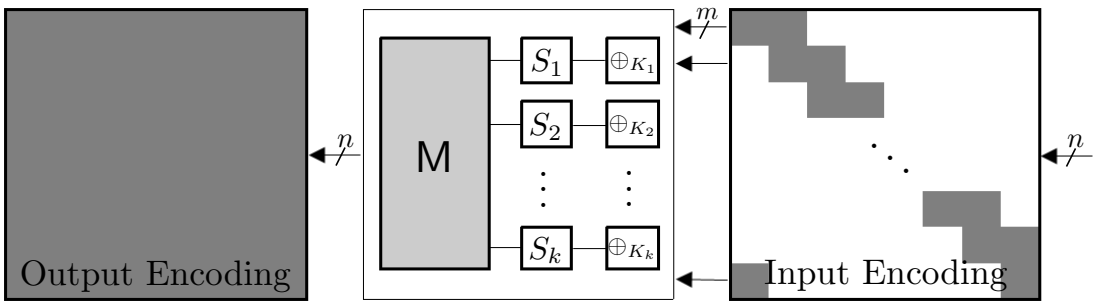

Fig. 5: The hard-to-invert encoded round function with $m_A = n$ and $\omega = 2$

However, when we have large m_A, it is not trivial to compute the inverse of F. Without using F^{-1} information, we have to guess about $\log m_A$ vectors, instead of one vector, to obtain m_A linearly independent vectors, which results in complexity

$$O\left(\tfrac{n}{m} \cdot {m_A}^3 2^{m(\log m_A + 2)}\right) = O\left(\tfrac{n}{m} \cdot {m_A}^{m+3} 2^{2m}\right) \tag{5}$$

for finding the affine encodings, which is a polynomial of m_A with degree $m + 3$ (much larger than 3 when the F^{-1} oracle is given).

Consider the following encoded round function F which has special "*sparse unsplit*" encodings as input encoding like in Fig 5: Let us consider an input encoding of the form $\oplus_a \circ A$ for $A \in \mathbb{Z}_2^{n\times n}$ and $a \in \mathbb{Z}_2^{n\times 1}$, where A is composed of k horizontal strips A_i of size $m\times n$ with $k = n/m$. We take A_i has zero columns at the j-th columns for $j \in \{1, 2, \cdots, n\}\backslash\{(i-1)m+1, \cdots, (i+\omega-1)m\}$ for some $\omega > 1$, *i.e.* the ωm columns of A_i are nonzero and the others are zero vectors. For $j \in \{(i-1)m+1, \cdots, (i+\omega-1)m\}$, it is considered as a number in $\{1, 2, \cdots, n\}$, which is congruent to j modulo n.

Then the encoded round function F can be expressed as a sum of F_i's such that

$$F(x_1, \cdots, x_k) = \textstyle\sum_{i=1}^{k-\omega+1} F_i(x_i, \cdots, x_{i+\omega-1}) + \sum_{i=k-\omega+2}^{k} F_i(x_i, \cdots, x_k, x_1, \cdots, x_{i+\omega-k-1})$$

for some $F_i : \{0,1\}^{\omega m} \to \{0,1\}^n$ since the input size related to a S-box S_i is ωm bits. Therefore, we can implement the encoded round function F composed of k lookup tables of $2^{\omega m}$-by-n, instead of constructing a huge table of 2^n-by-n.

However, $F_i(x, 0, \cdots, 0)$ can be considered as the bijection on m bits, which is affine equivalent to S_i. Then, if F_i is public, then the EAEA can be applied to each F_i and it has only $O(m^3 \cdot 2^{2m})$ complexity for each i. To prevent the individual attack, we use random functions f_i from ωm bits to n bits to modify the function F_i. If ω is an even integer, we can replace $F_i(x_i, \cdots, x_{i+\omega-1})$ by

$$F_i(x_i, \cdots, x_{i+\omega-1}) + f_i(x_i) + \cdots + f_{i+\omega-1}(x_{i+\omega-1}).$$

For the index j lager than k, we define x_j and f_j by x_{j-k} and f_{j-k}, respectively. This modification can express the encoded round function F as a sum of $2^{\omega m}$-by-n tables without revealing F_i's (For the case where ω is odd, see Appendix D.1). It needs the storage of $\frac{n}{m} \cdot n \cdot 2^{\omega m}$ bits in order to implement the F, which is 16 MB and 64 MB for $n = 128$ and 256, respectively, when $m = 8$ and $\omega = 2$. The detailed description of an instance for the case $\omega = 2$ can be found in Appendix D.2.

We could use the EAEA, by computing the inverse of F: Using meet-in-the-middle attack (MITM), one inverse evaluation of F takes $O(m_A 2^{m_A/2})$ time complexity and $O(m_A 2^{m_A/2})$ memory, which can be reduced to $O(m_A 2^{m_A/4})$ using a technique similar to the dissection [10] (See the Appendix E). Since the EAEA requires about $\log m_A$ number of evaluation of F^{-1}, its complexity is

$$O\left(\tfrac{n}{m} \cdot m_A^3 2^{3m} + \tfrac{n}{m_A} \cdot \log m_A \cdot m_A \cdot 2^{m_A/2}\right)$$

which is dominated by the inverting complexity when $m_A > 6m$. This complexity is exponential in m_A, but less than (5) when $m_A \leq 128$.

The hardness of inverting F which has *sparse unsplit* encodings can be considered as a special version of sparse subset sum problem (SSSP). The following form of the SSSP is introduced in [11] and used in [11, 8, 18] designing fully homomorphic encryptions: Let v, k be integers with $k << v$. Given a positive integer u, a list $(y_1, \cdots, y_v) \in \mathbb{Z}_u$, and another element $y \in \mathbb{Z}_u$ such that

$$y \equiv \sum_{i=1}^{v} \delta_i \cdot y_i \pmod{u}$$

where

$$\delta_i \in \{0,1\}, \sum_{i=1}^{\lfloor v/k \rfloor} \delta_i = \sum_{\lfloor v/k \rfloor + 1}^{2\lfloor v/k \rfloor} \delta_i \cdots = \sum_{i=(k-1)\lfloor v/k \rfloor + 1}^{v} \delta_i = 1,$$

the problem is to find the coefficients δ_i.

If $\omega = 1$, the inverting F can be regarded as this problem by embedding $\{0,1\}^n$ to $\mathbb{Z}_{2^n}$. In that case, this SSSP can be solved in time complexity $\widetilde{O}(2^{n/2})$ with $\widetilde{O}(2^{n/4})$ memory according to a variant of Schroeppel-Shamir algorithm [17]. It has the same complexity as the proposed MITM. For the case $\omega > 1$, since the m-bit value x_i is used the input value of several F_j's, the computing F^{-1} is is slightly different from this SSSP. Currently, we don't have any method to exploit this feature. To sum up the complexity for recovering affine encoding is

$$O\left(\min\left\{\tfrac{n}{m} \cdot n^{m+3} \cdot 2^{2m}, n \cdot \log n \cdot 2^{n/2}\right\}\right)$$

for $m_A = n \geq 128$.

Table 1 shows that the security and storage of the proposed implementation for the case $m_A = n, m = 8$ and $\omega = 2$ according to the block size n of block cipher. As seen in the Table 1, the attack complexity can be large up to 2^{104} and 2^{109} when $n = 192$ and 256, respectively. To design for the secure white-box implementation, a block cipher with larger input size is preferred. For example, Rijndael-128-192 [9], a 128-bit key, a 192-bit block cipher can be candidate of a based cipher of white-box implementation.

n	Security $\min\left\{\frac{n}{m} \cdot n^{m+3} \cdot 2^{2m}, n \cdot \log n \cdot 2^{n/2}\right\}$	Storage $\frac{n}{m} \cdot n \cdot 2^{2m}$ bits
128	$O(n \cdot \log n \cdot 2^{\frac{n}{2}}) = 2^{74}$	16 MB × (# of rounds)
192	$O(\frac{n}{m} \cdot n^{m+3} \cdot 2^{2m}) = 2^{104}$	36 MB × (# of rounds)
256	$O(\frac{n}{m} \cdot n^{m+3} \cdot 2^{2m}) = 2^{109}$	64 MB × (# of rounds)

Table 1: The security and storage of the proposed WB implementation for $m = 8$ and $\omega = 2$

5 Conclusion

In this paper, we proposed a general analytic toolbox of white-box implementation, which can efficiently extract the secret encodings obfuscated in implementation when its design follows our general model. With our toolbox, it is very easy to evaluate the asymptotic complexity for any white-box implementation in our general model and all previous designs belong to this model. Hence our toolbox could be used to give a security measure about white-box implementations.

Another advantage of our toolbox is that we can remove insecure design in early stage and concentrate on more plausible approaches. We showed that when the encoded round function is hard to invert, the complexity of the EAEA becomes significantly larger. We expect that this leads to an approach to design a practically secure white-box implementation.

References

1. E. Barkan and E. Biham. In How Many Ways Can You Write Rijndael? In Y. Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 160–175. Springer, 2002.
2. O. Billet, H. Gilbert, and C. Ech-Chatbi. Cryptanalysis of a White Box AES Implementation. In H. Handschuh and M. A. Hasan, editors, *Selected Areas in Cryptography - SAC 2004*, volume 3357 of *LNCS*, pages 227–240. Springer, 2005.
3. A. Biryukov, C. D. Canniére, A. Braeken, and B. Preneel. A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 33–50. Springer, 2003.
4. A. Biryukov and A. Shamir. Structural Cryptanalysis of SASAS. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 395–405. Springer, 2001.
5. J. Bringer, H. Chabanne, , and E. Dottax. White Box Cryptography: Another Attempt. `http://eprint.iacr.org`, 2006.
6. S. Chow, P. Eisen, H. Johnson, and P. C. V. Oorschot. White-Box Cryptography and an AES Implementation. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography - SAC 2002*, volume 2595 of *LNCS*, pages 250–270. Springer, 2003.
7. S. Chow, P. Eisen, H. Johnson, and P. C. van Oorschot. A White-Box DES Implementation for DRM Applications. In J. Feigenbaum, editor, *Digital Rights Management - DRM 2002*, volume 2696 of *LNCS*, pages 1–15. Springer, 2003.
8. J.-S. Coron, D. Naccache, and M. Tibouchi. Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer Berlin Heidelberg, 2012.
9. J. Daemen and V. Rijmen. AES Proposal: Rijndael, 1998.
10. I. Dinur, O. Dunkelman, N. Keller, and A. Shamir. Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial search Problems. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *LNCS*, pages 719–740. Springer, 2012.
11. C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 129–148. Springer-Verlag, 2011.
12. M. Karroumi. Protecting White-Box AES with Dual Ciphers. In K.-H. Rhee and D. Nyang, editors, *Information Security and Cryptology - ICISC 2010*, volume 6829 of *LNCS*, pages 278–291. Springer, 2011.
13. T. Lepoint, M. Rivain, Y. D. Mulder, P. Roelse, and B. Preneel. Two Attacks on a White-Box AES Implementation. In T. Lange, K. Lauter, and P. Lisoněk, editors, *Selected Areas in Cryptography - SAC 2013*, LNCS, pages 265–285. Springer, 2014.
14. W. Michiels, P. Gorissen, and H. D. L. Hollmann. Cryptanalysis of a Generic Class of White-Box Implementations. In R. M. Avanzi, L. Keliher, and F. Sica, editors, *Selected Areas in Cryptography - SAC 2008*, volume 5381 of *LNCS*, pages 414–428. Springer, 2009.
15. Y. D. Mulder, P. Roelse, and B. Preneel. Cryptanalysis of the Xiao - Lai White-Box AES Implementation. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography - SAC 2012*, volume 7707 of *LNCS*, pages 34–49. Springer, 2013.
16. Y. D. Mulder, B. Wyseur, and B. Preneel. Cryptanalysis of a Perturbated White-Box Aes Implementation. In G. Gong and K. C. Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 292–310. Springer, 2010.
17. R. Schroeppel and A. Shamir. A TcS2 = 0 (2n) time/space tradeoff for certain NP-complete problems. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 328–336, Oct 1979.
18. N. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.
19. B. Wyseur, W. Michiels, P. Gorissen, and B. Preneel. Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings. In C. Adams, A. Miri, and M. Wiener, editors, *Selected Areas in Cryptography - SAC 2007*, volume 4876 of *LNCS*, pages 264–277. Springer, 2007.
20. Y. Xiao and X. Lai. A Secure Implementation of White-box AES. In *Computer Science and its Applications - CSA 2009*, pages 1–6. IEEE, 2009.

A Examples of White-Box Implementations and cryptanalysis

In this section, we introduce examples of white-box implementations and explain them in sense of our generalized model defined in Section 2. Also, we provide brief illustration about how to apply our attack tools to the white-box implementations.

Chow et al.'s implementation Chow et al. presented a white-box AES implementation in [6]. They used encodings composed of nonlinear mappings and linear mappings. The subround function F over $\mathbb{F}_{2^8}^4$ of Chow et al.'s implementation is the form $F = QBMSAP$, where P, Q are concatenation of 4-bit nonlinear permutations, A, B are block diagonal linear mapping with block size 8, S is the bytewise operation of S-boxes, and M is the Mixcolumn operation of size 32. Note that AddRoundKey operation can be merged to the nonlinear encoding P. Since the block size of encodings is 8 which is the same as the input size of S-boxes, then ShiftRow operation can be omitted in round function. So, we consider the round function as block diagonal mapping with block size 32.

Note that they use input encodings of 8-bit, same input size of S-boxes, to maintain the number of input bits which affect S-boxes. Then the number of input bits of lookup tables is also 8 and so 8 is suitable size of the input encodings in a practical aspect. However, for the security of the implementation, it is not appropriate because the implementation used 8-bit input encodings are vulnerable to Billet et al.'s attack [2]. Billet et al. presented how to extract the AES secret key from Chow et al.'s white-box AES implementation when the input size of the encodings is same to the input size of the S-boxes. The complexity of recovering nonlinear parts is 2^{24} and the total complexity of recovering 128-bit AES key is less than 2^{30}. Furthermore, Michiels et al. [14] presented a cryptanalysis for white-box implementation of generic class of SLT ciphers when input encodings whose the input size is same to the input size of S-box are used in the implementation.

On the other hand, in our notations we can let $m = m_A = 8$ and $m_P = m_Q = 4$. Thus, applying the result in Section 3.1, we can recover the nonlinear encodings in $2k_Q \cdot 2^{3m_Q} = 2 \cdot 32 \cdot 2^{3 \cdot 4} = 2^{18}$ and this result is better than Billet et al.'s result $2 \cdot 16 \cdot 2^{3 \cdot 8} = 2^{29}$. The only thing to be careful about is to take multiset of plaintexts. We have to take multiset of plaintexts, so that the function is related to two S-boxes, for example, the multiset of plaintexts of size 128-bit values that has the property $\mathsf{C}_4\mathsf{P}_8\mathsf{C}_4^{29}$.

After removing the nonlinear parts of input/output encodings, one can obtain the bijection map F on $\mathbb{F}_{2^8}$ which is affine equivalence to the S-box by fixing input value on all bits but a byte. Then by the affine equivalence algorithm described in Section 3.2, input/output encodings can be recovered in $km^3 2^{2m} = 2^{29}$ step where $k = 16, m = 8$. The total complexity is less than 2^{30} and comparable to the Billet et al.'s result, 2^{30}.

Xiao and Lai's implementation In 2009, Xiao and Lai proposed a white-box AES implementation [20] which is secure against Billet et al.'s attack. They use only the linear mappings for input/output encoding. The input bit size of the input encodings is twice of the input bit size of the S-boxes. Since AddRoundkey operation is implemented before SubByte operation, we may assume that an input encodings are invertible affine mappings over $\mathbb{Z}_2^{16}$. Let an input encoding A be invertible affine mapping over $\mathbb{Z}_2^{16}$ and an output encoding B be 32×16 linear mapping with rank 16. Then, the subround function F from $\mathbb{Z}_2^{16}$ to $\mathbb{Z}_2^{32}$ in [20] is defined by $F = B \circ S \circ A$, where S is bytewise operation of S-boxes. Note that MixColumn operation is considered a part of output encoding B.

They used 16-bit input encodings in order to avoid Billet et al.'s attack. Hence the number of input bits which affect S-box increased and the storage of tables also increased. They expected the security of implementation also would increase, but it was not true because (1) they only used linear maps, not affine nor nonlinear parts and (2) they used block diagonal linear maps with small size, which are vulnerable to linear equivalence algorithm. Mulder et al. [15] presented a cryptanalysis of Xiao and Lai's implementation which extracts the secret key in complexity 2^{32} using linear equivalence algorithm of [3].

Now let's apply our attack tools to this implementation. By fixing input value on all bits but 2 bytes as a constant, one can obtain the bijection map F on $\mathbb{Z}_2^{16}$ as following form:

$$F = B \circ (S, S) \circ (\oplus_k', \oplus_k'') \circ A$$

where A, B are linear invertible maps on 16 bits. Then F is affine equivalent to (S, S). *i.e.* $m = 8$ and $n = 16$, in our notations.

Applying extended affine equivalence algorithm, we can recover the secret encodings in $\frac{n}{m}n^3 2^{2m} = 2^{29}$ steps. The total complexity of finding all A's and B's is $8 \cdot 2^{29} = 2^{32}$ and hence is comparable to the previous result in [15], 2^{32}.

Karroumi's implementation Karroumi presented a white-box AES implementation [12] using the dual representations of the AES cipher and design technique of Chow et al.'s white-box AES implementation. A cipher $\overline{\text{AES}}$ is called a dual cipher of AES if there exists an isomorphism Δ on $\mathbb{F}_{2^8}$ such that

$$\mathbf{\Delta}\left(\text{AES}_K(m)\right) = \overline{\text{AES}}_{\mathbf{\Delta}(K)}\left(\mathbf{\Delta}(m)\right), \quad \text{for all } K, m \in \mathbb{F}_{2^8}$$

where $\mathbf{\Delta} = (\Delta, \cdots, \Delta)$ is the bytewise operation of Δ. Dual ciphers of the AES was first presented by Barkan and Biham in 2002 [1]. In [1], they present several AES dual ciphers, which are equivalent to the original, considering modification of the polynomial representation of $\mathbb{F}_{2^8}$, the coefficients of the MixColumn operation, and the affine transformation in the SubByte function. Biryukov et al. [3] proposed that there are at least $61,200$ AES dual ciphers by expanding the result in [1]. With $61,200$ dual ciphers, Karroumi modify the algebraic operations in each AES round function such as SubBytes, MixColums, and AddKeyRounds. Then, the subround function F over $\mathbb{Z}_2^{32}$ of Karroumi's implementation is the form $F = QBMSAP$, where P, Q are concatenations of 4-bit nonlinear permutations, A, B are block diagonal linear mappings with block size 8, M is the mixing bijection of size 32, and S is a concatenation of S-boxes. For dual cipher, an automorphism Δ is a block diagonal affine mapping with block size 8 and it can be merged to the encodings.

As a result, Karroumi's implementation is of same form with the Chow et al.'s implementation. Thus, the total attack complexity is 2^{29}.

B Proofs of the Lemmas

B.1 Proof of Lemma 1

We may assume that A is linear, because an addition by a constant preserves property B_m when M has an even number of elements.

Let $A = \begin{bmatrix} A_1 & \cdots & A_n \end{bmatrix}$ with column vectors A_i's and $A^* = \begin{bmatrix} A_{i_1} & \cdots & A_{i_\tau} \end{bmatrix}$. Then we have $A(M) = \{A^*(x') + b \mid x' \in \pi_I(M)\}$ for some constant vector $b \in \mathbb{Z}_2^m$. It is enough to show that the multiset $A^*(\pi_I(M))$ has property B_m.

If $\tau = m$ and A^* has rank m, then the multiset $A^*(\pi_I(M))$ of m-bit values has property P_m and hence property B_m since $m > 1$. Otherwise, the size of the kernel of A^* is $2^{\tau - \text{rank}(A^*)}$ and hence the number of preimage of $y \in A^*(\pi_I(M))$ is $2^{\tau - \text{rank}(A^*)}$. Since $2^{\tau - \text{rank}(A^*)}$ is even, the multiset $A^*(\pi_I(M))$ has property E_m. It follows that $A^*(\pi_I(M))$ has property P_m or E_m. □

B.2 Proof of Lemma 2

Let L be the linear part of A and b be the constant part of A. Then

$$\sum_{y \in A(M)} y = \sum_{x \in M} (Lx + b) = L\left(\sum_{x \in M} x\right) + \sum_{x \in M} b = 0$$

since the multiset M has property B_n and the size of M is even. □

C The EAEA with split A

When we use the EAEA for the white-box implementation, we can reduce the complexity for the case where input encoding A is of some special form.

For convenience, we let A and B are linear. Let's consider $A \in (\mathbb{Z}_2)^{n\times n}$ as a $\tilde{A} \in (\mathbb{Z}_2^{m\times m})^{k\times k}$, where $n = km$. If $\tilde{A}$ is block-diagonal map, then we can perform separately the above attack on each block. If the size of the each block of block-diagonal map $\tilde{A}$ is k_i with $\sum_i k_i = k$, $A_i \in \mathbb{Z}_2^{k_i m\times k_i m}$ is i-th block of A and $B_i \in \mathbb{Z}_2^{n\times k_i m}$ is i-th vertical strip of B correspond to A_i, we can find maps of the form $F_i = B_i \circ (S||\cdots||S) \circ A_i$, where $(S||\cdots||S)$ is concatenation of k_i S-boxes. Since image of F_i has rank k_i, we can find a $k_i \times n$ matrix C_i satisfying that $C_i \circ F_i$ is bijective. Then we obtain bijective maps of the following form:

$$\tilde{F}_i = \tilde{B}_i \circ (S||\cdots||S) \circ A_i$$

where $\tilde{B}_i = C_i \circ B_i$. Thus we can recover the encodings in complexity $\sum_i k_i (k_i m)^3 2^{3m}$, less than $kn^3 2^{3m}$.

More generally, if $\tilde{A}$ can be split into two or more bijective map, that is, A is *split* as defined in section 3.2, we can apply the above argument to $\tilde{A}$. In detail, in the case that $(\tilde{A})_{i,j} = 0^{m\times m}$ for $(i,j) \in [k_1+1, k_1+k_0] \times ([1,n] \setminus [k_2+1, k_2+k_0])$ or $(i,j) \in ([1,n] \setminus [k_1+1, k_1+k_0]) \times [k_2+1, k_2+k_0]$ fore some k_0, k_1, k_2 with $k_1+k_0, k_2+k_0 \le k$,

i.e. $\tilde{A}$ is the form of $\left[\begin{array}{c|c|c} * & 0 & * \\ \hline 0 & A^* & 0 \\ \hline * & 0 & * \end{array}\right]$ one can obtain a bijective map on $\mathbb{Z}_2^{k_0 m}$ using small submatrix, A^* in the above.

D Our Design for the White-Box implementation

D.1 Representation for the Lookup Tables when ω=odd

Let $\omega > 1$ be an odd number. If we use $F_i(x_i, \cdots, x_{i+\omega-1}) + f_i(x_i) + \cdots + f_{i+\omega-1}(x_{i+\omega-1})$ instead of $F_i(x_i, \cdots, x_{i+\omega-1})$ as the even case, then

$$\begin{aligned}&\sum_{i=1}^{k} \Big(F_i(x_i, \cdots, x_{i+\omega-1}) + f_i(x_i) + \cdots + f_{i+\omega-1}(x_{i+\omega-1})\Big)\\ &= F(x) + f_1(x_1) + \cdots + f_k(x_k) \neq F(x)\end{aligned}$$

because ω is odd. (the notation for the indices follows Section 4)

Hence, We use more functions g_i's and h_i's. For $i = 1, \ldots, k$, we define g_i by a random function from ωm bits to n bits and $h_i = f_i + g_i$ from ωm bits to n bits. Then we can replace F_i with $F_i(x_i, \cdots, x_{i+\omega-1})$ by

$$F_i(x_i, \cdots, x_{i+\omega-1}) + f_i(x_i) + \cdots + f_{i+\omega-3}(x_{i+\omega-3}) + g_{i+\omega-2}(x_{i+\omega-2}) + h_{i+\omega-1}(x_{i+\omega-1}),$$

since the sum of these functions is equal to $F(x)$. Therefore, we can express the encoded round function F as a sum of $2^{\omega m}$-by-n tables without revealing F_i's for all cases ω.

D.2 An instance of Our Design for the White-Box implementation ($\omega = 2$)

In this subsection, we propose a white-box implementation of SLT ciphers based on the above approach. We use the *sparse unsplit* encodings as input encodings like in Fig 5 to reduce the storage size. As seen in the Fig 5, we only consider the case that $m_A = n$ and $\omega = 2$.

Let us consider the round function $E = M \circ S \circ \oplus_K$ of SLT cipher with block size n, where M is a linear mapping on n bits, S is a concatenation of S-boxes $S_1, \cdots, S_k$ on m bits, and $K = (K_1, \cdots, K_k)$

is an round key of n-bit with $n = km$. Let A be a linear bijection on n bits and a be a n-bit value vector. We consider the affine bijection $\oplus a \circ A$ as an input encoding. For an input encoding A' of the next round, we define output encoding B by $B = (A')^{-1} \circ M$. Then, the encoded round function F of E is defined by $F = B \circ S \circ \oplus_{\bar{K}} \circ A$, where $\bar{K} = K + a$.

Consider $A \in \mathbb{Z}_2^{n \times n}$ to be partitioned into k^2 blocks $A_{i,j}$ of size $m \times m$ with $k = n/m$. For $i = 1, \cdots, k-1$, we take $A_{i,j}$ is zero matrix for all $j \neq i, i+1$. If $i = k$, we take $A_{k,j}$ is zero matrix for all $j \neq 1, k$. That is,

$$(A_{i,j})_{x,y} = A_{(i-1)m+x,(j-1)m+y}$$

so that

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \mathbf{0} & \mathbf{0} \cdots & \mathbf{0} \\ \mathbf{0} & A_{2,2} & A_{2,3} & \mathbf{0} \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \ddots & \vdots \\ A_{k,1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \cdots & A_{k,k} \end{bmatrix}.$$

Let B_i be an affine function from m-bit to n-bit such that $B(x) = B_1(x_1) + \cdots + B_k(x_k)$ for $x = (x_1, \cdots, x_k) \in \{0,1\}^n$ where x_i is m-bit value. Now, we define $F_i : \{0,1\}^{2m} \to \{0,1\}^n$ by $F_i(x, y) = B_i \circ S_i(A_{i,i}(x) + A_{i,i+1}(y) + \bar{K}_i)$ for $x, y \in \{0,1\}^m$, where $\bar{K} = (\bar{K}_1, \cdots, \bar{K}_k)$. Then, the encoded round function F can be expressed as a sum of F_i's:

$$F(x_1, x_2, \cdots, x_k) = F_1(x_1, x_2) + F_2(x_2, x_3) + \cdots + F_k(x_k, x_1).$$

Therefore, we can implement the encoded round function F composed of k lookup tables of 2^{2m}-by-n, instead of constructing a huge table of 2^n-by-n.

However, since $F_i(x, 0) = B_i \circ S_i \circ \oplus_{\bar{K}_i} \circ A_{i,i}(x)$ for m-bit value x, $F_i(x, 0)$ can be considered as the bijection on m bits, which is affine equivalent to S_i. Then, the EAEA can be applied to each F_i and it has only $O(m^3 \cdot 2^{2m})$ complexity for each i.. To prevent the individual attack, we use random functions f_i to modify the function F_i. We replace $F_i(x_i, x_{i+1})$ by $T_i : \{0,1\}^{2m} \to \{0,1\}^m$ such that

$$T_i(x, y) = \begin{cases} F_i(x, y) + f_i(x) + f_{i+1}(y), & \text{if } i \neq k \\ F_k(x, y) + f_k(x) + f_1(y), & \text{if } i = k \end{cases}$$

for any random function f_i from m-bit to n-bit. If we set $x_{k+1} := x_1$,

$$\sum_{i=1}^{k} T_i(x_i, x_{i+1}) = \sum_{i=1}^{k} F_i(x_i, x_{i+1}) = F(x_1, \cdots, x_k)$$

for $x = (x_1, \cdots, x_k)$ is n-bit value.

Therefore, we can express the encoded round function F as a sum of 2^{2m}-by-n tables without revealing F_i's. The required storage is $\frac{n}{m} \cdot n \cdot 2^{2m}$ bits, which is 16 MB, 36 MB, and 64 MB for $n = 128, 192, 256$, respectively, when $m = 8$. Since f_i is an random function which has no certain structure unlike F_i, we cannot efficiently extract encodings from T_i's. The effect of f_i can be only removed after T_i's are combined in F, because f_i is circularly blended into F_{i-1} and F_i. In that case $m_A = n$, the proposed implementation has security

$$O\left(\min\left\{\frac{n}{m} \cdot {m_A}^{m+3} \cdot 2^{2m}, n \cdot \log m_A \cdot 2^{m_A/2}\right\}\right) = \begin{cases} O(n \cdot \log n \cdot 2^{\frac{n}{2}}) = 2^{74}, & \text{if } n = 128 \\ O(\frac{n}{m} \cdot n^{m+3} \cdot 2^{2m}) = 2^{104}, & \text{if } n = 192 \\ O(\frac{n}{m} \cdot n^{m+3} \cdot 2^{2m}) = 2^{109}, & \text{if } n = 256 \end{cases}.$$

E Computing Inverse of F

In the white-box implementation, the encoded round function F consists of several sub-functions as in Section 2.2. In particular, if the input/output encodings are affine, F can be expressed as $F = \sum_j F_j$. Because F_j's input/output values are given in the tables, we can easily access to the inverse of F_j. However, it is not easy to compute the inverse of F when F is the sum of the F_j's. In order to use the EAEA, several number of evaluation of F^{-1} are required, so we need to check the complexity to compute the inverse of F.

As in Section 3.2, we let F be a bijection on n bits. For simplicity, we assume $F(x) = \sum_{j=1}^{k} F_j(x_j)$ for $F_j : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$, where $x = (x_1, \cdots, x_k)$ and x_j's are m-bit values with $n = km$. A trivial approach to invert F is the exhaustive search, which takes 2^n time complexity. One can improve it using the meet-in-the-middle (MITM) attack: By combining functions, we let $F(x) = G_1(x_1, \cdots x_{\lfloor \frac{k}{2} \rfloor}) + G_2(x_{\lfloor \frac{k}{2} \rfloor + 1}, \cdots, x_k)$. For $y \in \mathbb{Z}_2^n$, one can make a table of $\oplus_y \circ G_1$, sort it by the output values, and compare it with the value of G_2. Then one can evaluate $F^{-1}(y)$ in $O(n2^{n/2})$ time complexity with $O(n2^{n/2})$ memory. The size of required memory for the MITM attack is quite large to implement - for example, 2^{38} GB are required for $n = 128$. We provide another method requiring smaller memory while maintaining asymptotic time complexity.

For convenience of notations, we let $F(x) = \sum_{j=1}^{4} F_j(x_j)$ where $x = (x_1, x_2, x_3, x_4)$ and each x_j is an m_0-bit value with $m_0 = \frac{n}{4}$. For a function f whose value is on n bits, $\tilde{f}$ denotes the projection of f on the first m_0 bits. To evaluate $F^{-1}(y)$ for any n-bit value y, we perform the following steps:

1. Guess m_0-bit value $\tilde{z}$ for $\tilde{f}_1(x_1) + \tilde{f}_2(x_2)$.
2. Perform the MITM attack using $\tilde{f}_1(x_1) + \tilde{f}_2(x_2) = \tilde{z}$ and store the list
$$L = \left\{ (x_1, x_2, f_1(x_1) + f_2(x_2) + y) \mid \tilde{f}_1(x_1) + \tilde{f}_2(x_2) = \tilde{z} \right\}$$
for the result of the MITM attack.
3. Perform the MITM attack using $\tilde{f}_3(x_3) + \tilde{f}_4(x_4) = \tilde{y} + \tilde{z}$, where $\tilde{y}$ is the first m_0-bit of y.
4. For each (x_3, x_4) satisfying $\tilde{f}_3(x_3) + \tilde{f}_4(x_4) = \tilde{y} + \tilde{z}$, compare $f_3(x_3) + f_4(x_4)$ with the values in L.

Both the average number of elements in L and the number of (x_3, x_4) satisfying $\tilde{f}_3(x_3) + \tilde{f}_4(x_4) = \tilde{y} + \tilde{z}$ are $2^{n/4}$. For one guessing of $\frac{n}{4}$-bit value, we perform 3 times of MITM attacks on the sets with $2^{n/4}$ cardinality. Therefore, we can evaluate $F^{-1}(y)$ in $O(n2^{n/2})$ time complexity with $O(n2^{n/4})$ memory. For the case of $n = 128$, the required memory is 64 GB, which is practical to implement.