An Efficient Affine Equivalence Algorithm for Multiple S-Boxes and a Structured Affine Layer

Jung Hee Cheon[1], Hyunsook Hong[1], Joohee Lee[1(✉)], and Jooyoung Lee[2]

[1] Seoul National University (SNU), Seoul, Republic of Korea
{jhcheon,hongsuk07,skfro6360}@snu.ac.kr
[2] KAIST, Daejeon, Republic of Korea
hicalf@gmail.com

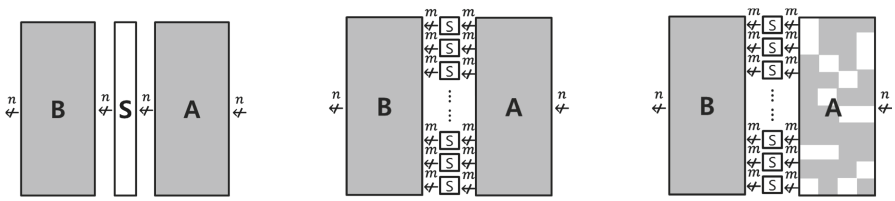
Abstract. An affine equivalence problem is to find affine mappings A and B such that $F = B \circ S \circ A$ for given two permutations F and S, which was first studied by Biryukov et al. Their algorithm for solving an affine equivalence problem is quite efficient and has been used in the cryptanalytic toolbox for many cryptographic schemes. Recently, Baek et al. presented a specialized affine equivalence algorithm (SAEA), which solves an affine equivalence problem in the case that S is a concatenation of several smaller S-boxes. The SAEA is more efficient than the affine equivalence algorithm for special cases, but its complexity mainly depends on the entire input size of F.

In this paper, we revisit the affine equivalence problem for a special ASA structure with multiple S-boxes and a structured input affine layer. We show that the work factor in SAEA can be reduced if the input affine layer in ASA has a certain structure. Moreover, the complexity of our algorithm mainly depends on the input size of smaller S-boxes, and not on the entire input size of F. We also present a new attack algorithm on the white-box AES implementation proposed by Baek et al. The cryptanalysis efficiently extracts the secret key from the implementation with a complexity of 2^{33}, where the claimed security level is 2^{110}.

Keywords: Affine equivalence algorithm · ASA structure · Multiple S-boxes · Structured affine mapping · White-box implementation

1 Introduction

In 1997, Even and Mansour [9] showed that for the independent n-bit keys K and K' and the random permutation P, the block cipher $E_{(K,K')}(x) = P(x \oplus K) \oplus K'$ is secure against an adversary with up to $\mathcal{O}(2^{n/2})$ queries. This block cipher, often referred to as the Even-Mansour cipher, is regarded as a minimal block cipher construction [8]. The three-layer scheme $E_{(A,B)}(x) = B \circ S \circ A(x)$ for which S is a substitution layer and B and A are secret affine mappings is a generalization of the Even-Mansour cipher, say ASA structure or three-layer scheme ASA. The problem of finding the affine layers for a given three-layer scheme ASA with a known S can be seen as the *affine equivalence problem*, which was introduced in [4].



(a) General ASA (b) Multiple S-Boxes (c) Multiple S-Boxes and Structured A

Fig. 1. Variants of the ASA structure

More precisely, the *affine equivalence problem* is to find the affine mappings A and B satisfying $F = B \circ S \circ A$ for two given permutations F and S of n bits, if they exist, as in Fig. 1(a). Biryukov et al. [4] proposed an algorithm, which solves the affine equivalence problem with a complexity of $O(n^3 2^{2n})$. Their algorithm is quite efficient and has been used as a cryptanalytic tool [11–16] for many cryptographic schemes. A variant of this problem appears in the white-box implementations, where the middle layer S consists of a concatenation of several m-bit S-boxes as in Fig. 1(b). Baek et al. [1] presented a specialized affine equivalence algorithm (SAEA), which solves the affine equivalence problem in this case. They showed that an ASA structure with multiple S-boxes requires $O\left(\min\left\{(n^{m+4}/m)2^{2m}, (n^4/m)2^{3m} + n\log n \cdot 2^{n/2}\right\}\right)$ steps to recover the secret affine mappings under the previous attacks.

In this paper, we propose an efficient attack algorithm for the special ASA structure with multiple S-boxes and a *structured* input affine layer. Especially, we consider a variant of the affine equivalence problem depicted as in Fig. 1(c) where S is a concatenation of m-bit S-boxes for $m = n/s$ and A is an $s \times s$ block matrix with $m \times m$ matrix entries which are zeros in at least one position of each row except one. Our algorithm has a complexity that mainly depends on the size of the smaller S-boxes, and not the entire input/output size of F. Furthermore, the main factor of the complexity of our algorithm related to n drops from n^{m+3} to n^3 compared to SAEA. In Table 1, we precisely compare our affine equivalence algorithm to previous results [1,4].

Table 1. Comparison to previous affine equivalence algorithms

Algorithm	Complexity (dominant part)
Naive approach	$n^3 2^{n^2+n}$
Affine equivalence algorithm [4]	$n^3 2^{2n}$
SAEA [1]	$\min\left\{\frac{n}{m} \cdot n^{m+3}2^{2m}, \ \frac{n}{m} \cdot n^3 2^{3m} + n\log n \cdot 2^{n/2}\right\}$
Our algorithm	$5\left(\frac{n}{m}\log\frac{n}{m}\right)n^3 + 5n^2 2^m + nm^2 2^{2m}$

m: the input size of smaller S-boxes in the S-layer of ASA sturcture (in [4], $m = n$)
n: an entire input/output size of the instance functions
The "naive approach" is to check if $B = F \circ A^{-1} \circ S^{-1}$ is affine and invertible for all As

Application to White-Box Implementations. A white-box implementation aims to obfuscate the secret key inside a cryptographic algorithm itself [6]. It is a way of implementing a cryptographic algorithm with a specialized attack model, thereby protecting the secret keys even in the situation that the adversary has a full access to the implementation of the cryptosystem and full control over its execution platform.

Given n-bit block ciphers as in [2,7], a naive approach to hide the secret key in such situations is to provide an input/output table of the original cipher with the secret key. However, this is not a practical solution since it is too heavy, e.g. It needs about 2^{102} GB for $n = 128$. To reduce storage requirements, the most popular approach is to decompose a cipher into round functions and split each round function as a sum of small tables [1,5,6,10,18]. Since the secret key can be easily exposed from the input/output behaviors of the round function, the table representations of round functions need to be obfuscated by secret encoding functions.

To obfuscate the secret key efficiently, the composition of an affine layer and a substitution layer with tiny S-boxes was usually considered as a secret encoding (SA as an output encoding and AS as an input encoding). Baek et al. [1] showed that composing the substitution layers of tiny S-boxes to the input/output encodings does not help to improve the security of the white-box implementations. Hence, the secret encodings would be reduced up to affine layers so that encoded round functions may have the ASA structure. One approach to split the table of ASA structure into smaller ones is to use an affine map whose linear part is a block diagonal matrix of $m \times m$ blocks as an input A layer, where m is the size of S-boxes. In this case, we can express the three layer scheme ASA as a sum of 2^m-by-n tables. However, this type of construction allows the block-wise attacks with the affine equivalence algorithm in [4], which results in a low complexity depending on the block size.

Recently, Baek et al. [1] proposed a white-box AES implementation (referred to as the BCH implementation) that uses the special input affine encoding with sparse non-zero $m \times m$ blocks which is depicted in Fig. 2. They made a point of trade-off between the above approach and a naive approach (to store an entire input/output table) to hide the secret key into the ASA structure and suggested a method for constructing the look-up tables of the encoded round functions with this special input affine encodings. The encoded round function in their implementation can be expressed as a sum of 2^{2m}-by-n tables instead of the 2^n-by-n table in the naive approach.

By the way, the affine input encodings in the BCH implementation exactly have a structure that we define. Applying our attack algorithm, we can efficiently extract the secret round key in the implementation with a complexity of 2^{33} for the case that the input size of the encoded round function is 256 bits, where the claimed security level is 2^{110}. We provide the attack complexities for the other parameters in the BCH implementation in Table 2. In future works, our attack algorithm for the special ASA would be a useful attack tool for white-box implementations.

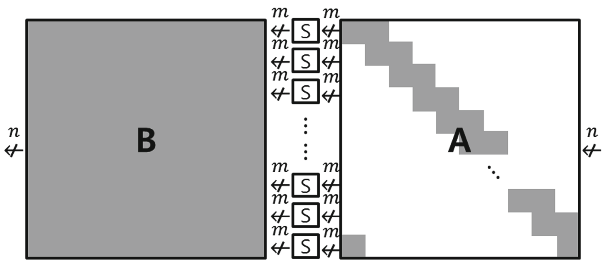


Fig. 2. The special structure lying in the input A layers of the BCH implementation

Table 2. The security of the BCH implementation, where n is the block size of encoded round function

n	Claimed security level in [1]	New security level
128	75 bits	32 bits
256	110 bits	33 bits
384	117 bits	34 bits

Outline of the Paper: In Sect. 2, we give some preliminaries used in this paper. Our attack for the special ASA structure is presented in Sect. 3. We give a cryptanalysis of the BCH implementation in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 Preliminaries

2.1 Structured Matrix

Fix parameters n, m, s such that $n = s \cdot m$ (throughout this paper), and we will consider an n-bit ASA scheme

$$F = B \circ S \circ A$$

such that the inner S-box S is given as a concatenation of s S-boxes of m-bit input/output size. We will also give a certain condition on the linear part L of A: when L is viewed as an $s \times s$ block matrix of $m \times m$ blocks, each row contains some zero entries except one row. The motivation of this particular structure is that such a scheme allows an efficient white-box implementation based on table look-ups. The block-wise density of a matrix can be represented by its block representing matrix, as defined as follows.

Definition 1 (Block Representing Matrix). *Let n, m, s be integers such that $n = s \cdot m$, and let L be an $n \times n$ matrix that is represented by a block matrix as follows.*

$$L = \begin{bmatrix} L_{1,1} & L_{1,2} & \cdots & L_{1,s} \\ L_{2,1} & L_{2,2} & \cdots & L_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ L_{s,1} & L_{s,2} & \cdots & L_{s,s} \end{bmatrix}$$

where $L_{i,j}$ is an $m \times m$ matrix for every i and j. Then the block representing matrix *of L, denoted by B_L, is defined as a binary $s \times s$ matrix where the (i,j)-entry is* 0 *if $L_{i,j}$ is the zero matrix and* 1 *otherwise.*

Definition 2 (Structured Matrix). *Let n, m, s be integers such that $n = s \cdot m$. A matrix L is called* structured *with respect to the block length m if L is invertible and the rows of its block representing matrix B_L are pairwise distinct.*

Example 1. The MixColumn step of AES-128 can be represented by a 128×128 matrix, say MC. When it is partitioned into 8×8 blocks, its 16×16 block representing matrix becomes

$$\mathsf{B}_{\mathsf{MC}} = \left[\begin{array}{cccc|cccc|cccc|cccc}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
\hline
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0
\end{array}\right].$$

Since MC is invertible over $\mathbb{F}_2$ and any two rows of above matrix B_{MC} are pairwise distinct, MC is structured.

An affine mapping A that maps n bits to n bits can be decomposed into a linear part L and a constant translation C as follows:

$$A(x) = L \cdot x + C$$

where L is an $n \times n$ matrix and C is an $n \times 1$ matrix over $\mathbb{F}_2$. We will say A is *structured* with respect to the block size m if the linear part L is structured with respect to the block size m.

2.2 Notation

We would set our notation used in Sects. 3 and 4. Throughout this paper, we set our target as a three-layer scheme $F = B \circ S \circ A$ of n bits which consists of a substitution and affine transformations. Our attack considers the case that the S layer contains s invertible S-boxes $S_1, S_2, \cdots, S_s$ of m bits, the output affine layer B is invertible, and the input affine layer A is structured. For the affine mappings A and B, we use the notation L and M to represent the linear part of A and B, and C and D to represent to the constant part of A and B, respectively. *i.e.,* The affine functions A and B are represented as follows:

$$A(x) = L \cdot x + C \text{ and } B(x) = M \cdot x + D$$

We consider the linear part L of A to be partitioned into s^2 $m \times m$ blocks. The (i, j)-th block matrix of size $m \times m$ is denoted by $L_{i,j}$. *i.e.,*

$$L = \begin{bmatrix} L_{1,1} & L_{1,2} & \cdots & L_{1,s} \\ L_{2,1} & L_{2,2} & \cdots & L_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ L_{s,1} & L_{s,2} & \cdots & L_{s,s} \end{bmatrix}$$

The linear part M of B can be partitioned into s vertical strips of size $n \times m$. We denote the i-th strip by M_i so that

$$M = \left[M_1 \middle| M_2 \middle| \cdots \middle| M_s \right]$$

For an arbitrary rectangular matrix N, we use a notation $\mathsf{col}(N)$ to represent the column space of N, namely a subspace of $\mathbb{F}_2^n$ spanned by the columns of N. We write the operation '+' to denote the bitwise XOR operation. We define $\oplus_K$ as the map $\oplus(x) = x + K$. Using this notation, we represent the key additions in a block cipher. We also split the n-bit string x into s m-bit blocks and write it as $x = (x_1, \cdots, x_s)$.

2.3 Our Problem Related to the Affine Equivalence Problem

We will formulate a problem, namely *specialized affine equivalence problem.* It can be regarded as a special variant of the affine equivalence problem. So, we first present the problem definition of the affine equivalence problem defined in [4] and then our problem related to the affine equivalence problem.

Given two permutations F and S, we say that F and S are *affine equivalent* if there exist invertible affine mappings A and B such that $F = B \circ S \circ A$. The *affine equivalence problem* is to find such affine mappings if they exist, by making a certain number of oracle queries to F and S.

We also take an attacker who can make oracle queries to F into account. The goal of this attacker might be to recover the affine layers with the knowledge of the three-layer scheme structure and input/output tables of m-bit S-boxes.

Definition 3 (Specialized Affine Equivalence Problem). *Consider a three-layer invertible ASA scheme $F = B \circ S \circ A$ of n-bit for which S is a concatenation of m-bit S-boxes and A is structured with respect to the block size*

m. We assume that the s m-bit S-boxes are given as input/output tables, and the block representing matrix of A with respect to the block length m is known. By making a certain number of oracle queries to F, we want to recover affine mappings A' and B' which are equivalent to A and B in the sense that:

- $F = B' \circ S \circ A'$
- *The block representing matrices of A and A' with respect to the block length m are the same.*

We can erase the assumption that m-bit S-boxes are given as tables. Then, we need to allow the oracle queries to S and store $sm2^m$ bits of input/output pairs of S-boxes in our algorithm in Sect. 3. We added an assumption that the block representing matrix of A with respect to the block length m is known since we can easily retrieve it with input/output behaviors of F in a practical scheme or it would be contained in an algorithm of a practical scheme, e.g. BCH implementation [1].

2.4 Useful Lemmas

In this subsection, we introduce useful lemmas which are used in our cryptanalysis.

Affine Equivalence Algorithm. Biryukov et al. [4] proposed an *affine equivalence algorithm* that efficiently solves the affine equivalence problem compared to the exhaustive search for A and B. The following lemma summarizes their result in terms of the complexity of the algorithm.

Lemma 1. *Let S_1 and S_2 be m-bit permutations. If S_1 and S_2 are affine equivalent, one can find all the pairs of affine mappings A and B such that $S_2 = B \circ S_1 \circ A$ in time $O(m^3 2^{2m})$.*

Rank of a Random Matrix over $\mathbb{F}_2$. The following lemma presented by Wan [17] tells us the property of random binary matrices.

Lemma 2. *Let n, k, r be integers such that $1 \le r \le \min(n, k)$. The probability that a random $n \times k$ binary matrix has rank r over $\mathbb{F}_2$ is*

$$P(n, k, r) = \frac{1}{2^{(n-r)(k-r)}} \cdot \prod_{i=0}^{r-1} \frac{(1 - 2^{i-k})(1 - 2^{i-n})}{(1 - 2^{i-r})}.$$

By Lemma 2, the simulation result shows that the probability that a random $n \times k$ binary matrix has rank $r \ge k - 5$ is greater than or equal to 0.99 for $n \le 1000$.

Affine Self-equivalences in Rijndael. The affine equivalence problem can have many equivalent solutions. For a permutation $\hat{S}$, if there exists nontrivial affine mappings a, b such that $\hat{S} = b \circ \hat{S} \circ a$, then we say that (a, b) is a self-equivalence of $\hat{S}$. The following lemma proposed by Biryukov et al. [4] tells us the number of affine self-equivalence of the S-box used in Rijndael [7].

Lemma 3. *The S-box $\hat{S}$ used in Rijndael has 2040 affine self-equivalences. In other words, there exists 2040 pairs of affine mappings (a, b) such that $\hat{S} = b \circ \hat{S} \circ a$.*

Intersection of Subspaces. For given two subspaces of $\mathbb{F}_2^n$, a complexity for computing an intersection of these two subspaces is less than $5n^3$ and is more precisely presented as follows.

Lemma 4. *For $0 < m_1 < m_2 < n$, suppose that V and W are subspaces of $\mathbb{F}_2^n$ of dimensions m_1 and m_2, respectively. For given bases of V and W, we can compute a basis for a subspace*

$$V \cap W$$

over $\mathbb{F}_2$ in a complexity of $n(2m_1^2 + 2m_1m_2 + m_2^2)$.

Proof. To calculate an intersection, consider the basis matrices $\bar{V}$ and $\bar{W}$ for V and W, respectively. Since

$$\bar{V} \cdot x = \bar{W} \cdot y \qquad \Longleftrightarrow \qquad [\bar{V}|\bar{W}] \cdot \begin{bmatrix} x \\ -y \end{bmatrix} = \mathbf{0},$$

we need to find the null space of $[\bar{V}|\bar{W}]$ with a Gaussian elimination in $n(m_1 + m_2)^2$ steps and then multiply $\bar{V}$ to the x's to obtain a basis for $V \cap W$ in less than nm_1^2 steps. □

3 Cryptanalysis of the ASA Structure with a Structured Affine Layer

In this section, we present an efficient algorithm solving the specialized affine equivalence problem defined in Definition 3. To avoid an abuse of notation, we first describe an instance of our algorithm for the specific cases which can be directly applied to the BCH implementation and then present a theorem for the general cases.

For an ASA structure $F = B \circ S \circ A$ whose notation is defined in Sect. 2.2, we would specify a class of L by defining its block representing matrix B_L with respect to block length m as follows.

$$(\mathsf{B}_L)_{i,j} = \begin{cases} 1, & \text{if } 1 \le i \le s - \beta + 1 \text{ and } i \le j \le i + \beta - 1 \\ 1, & \text{if } s - \beta + 1 < i \le s \text{ and } 1 \le j \le i + \beta - s - 1 \\ 0, & \text{otherwise} \end{cases},$$

for some positive integer $\beta < \left\lfloor \frac{s}{2} \right\rfloor$.

In other words, the $s \times s$ block representing matrix B_L of L would be depicted as:

$$\mathsf{B}_L = \begin{bmatrix} 1 & 1 & \cdots & 1 & & & & & \\ & 1 & 1 & \cdots & 1 & & & & \\ & & 1 & 1 & \cdots & 1 & & & \\ & & & \ddots & \ddots & \ddots & \ddots & & \\ & & & & 1 & 1 & \cdots & 1 & \\ 1 & & & & & 1 & \cdots & 1 & \\ \vdots & \ddots & & & & & \ddots & \vdots & \\ 1 & \cdots & 1 & & & & & 1 & \end{bmatrix}, \quad (1)$$

where each row and column contains β nonzero entries. Note that all the rows of B_L are distinct so that L is structured.

Summary of Our Approach. Our cryptanalysis is divided into two phases. Before we start to describe our attack, we summarize our cryptanalysis as below.

Phase 1. We first find the column spaces $\mathsf{col}(M_i)$ for all $1 \le i \le s$. Then, we can recover the linear part of output affine layer B up to a block diagonal matrix of block size m. Though we cannot obtain the exact M, it is an essential step to reduce output sizes of F from n to m.

Phase 2. From the phase 1, we can split F into $\tilde{F}_i$ for $1 \le i \le s$ which are the ASA structures from βm bits to m bits, respectively. We transform $\tilde{F}_i$ into an invertible ASA structure on m bits reducing the input sizes from βm to m. Then, the affine equivalence algorithm can be applied to the invertible ASA structure on m bits.

3.1 Decomposing the Linear Part of B

The first phase of our attack is to recover the linear part of B upto a block diagonal matrix. For each index $1 \le i \le s$, we will choose a certain number of pairs of plaintexts (P_1, P_2) having a difference only in the i-th m-bit block. Namely, when we write

$$P_1 = (x_1, x_2, \cdots, x_i, \cdots, x_s)$$
$$P_2 = (y_1, y_2, \cdots, y_i, \cdots, y_s)$$

for m-bit blocks x_j and y_j, $j = 1, \ldots, s$, we have $x_j = y_j$ for every $j \ne i$, but $x_i \ne y_i$. For any of such pairs (P_1, P_2), $S \circ A(P_1)$ and $S \circ A(P_2)$ will have non-zero differences exactly in β blocks since each column of B_L contains β 1's and S is defined as a concatenation of m-bit S-boxes. Specifically, we have

$$S \circ A(P_1) + S \circ A(P_2) = (\Delta_1, \cdots, \Delta_s),$$

where $\Delta_{i-\beta+1}, \cdots, \Delta_i$ are all non-zero blocks and the others are all zero blocks (cyclically indexed modulo s). So the positions of non-zero blocks are cyclically shifted as the index i increases. Since

$$F(P_1) + F(P_2) = B \circ S \circ A(P_1) + B \circ S \circ A(P_2) = M \cdot (S \circ A(P_1) + S \circ A(P_2))$$

$F(P_1) + F(P_2)$ would be always a linear combination of the βm columns from $M_{i-\beta+1}$ to M_i, namely

$$F(P_1) + F(P_2) \in \mathsf{col}(M_{i-\beta+1}|M_{i-\beta+2}|\cdots|M_i).$$

In order to find the column space $\mathsf{col}(M_{i-\beta+1}|M_{i-\beta+2}|\cdots|M_i)$, we set P_1+P_2 to have nonzero entries exactly in β blocks and compute $F(P_1) + F(P_2)$ for random P_1's in $\{0,1\}^n$ to collect βm linearly independent vectors over $\mathbb{F}_2$. Note that the probability that a random $n \times (\beta m+5)$ binary matrix has rank $r \geq \beta m$ is greater than 0.99 when $n \leq 1000$ by Lemma 2. Hence, from $\beta m + 5$ vectors of the form $F(P_1) + F(P_2)$, we can find the basis of this column space with a high probability($\geq$0.99) via the Gaussian elimination which takes $n(\beta m + 5)^2$ time. Since M is invertible over $\mathbb{F}_2$ and $\beta < \left\lfloor \frac{s}{2} \right\rfloor$, we have

$$\mathsf{col}(M_i) = \mathsf{col}(M_{i-\beta+1}|M_{i-\beta+2}|\cdots|M_i) \cap \mathsf{col}(M_i|M_{i+1}|\cdots|M_{i+\beta-1}).$$

Therefore we can compute a basis of $\mathsf{col}(M_i)$ in $5n(\beta m)^2$ time by Lemma 4. Overall, this phase requires $sn[(\beta m+5)^2+5(\beta m)^2]$ time complexity and $2s(\beta m+5)$ chosen plaintexts.

Now, we obtained the basis of each space $\mathsf{col}(M_i)$ for $1 \leq i \leq s$. Let $\tilde{M}_i \in \mathbb{F}_2^{n\times m}$ denote the matrix whose columns are the basis of $\mathsf{col}(M_i)$. Then each column of M_i can be represented by a linear combination of the columns of $\tilde{M}_i$ with certain unknown coefficients. So we have a decomposition as follows.

$$M = \tilde{M} \cdot U$$

where

$$\tilde{M} = \left[\tilde{M}_1 \middle| \tilde{M}_2 \middle| \cdots \middle| \tilde{M}_s\right] \quad \text{and} \quad U = \begin{bmatrix} U_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & U_2 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & U_3 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & U_s \end{bmatrix}$$

for some (unknown) $m \times m$ invertible matrices $U_1, \ldots, U_s$.

3.2 Recovering *A* and *B*

The second phase is to split the entire structure F on n bits into smaller ASA structures on m bits, and then apply the affine equivalence algorithm given in Lemma 1 to each of the smaller structures.

Let $\tilde{F}$ be a map defined by $\tilde{F}(X) = \tilde{M}^{-1} \cdot F(X)$ for every $X \in \mathbb{F}_2^n$. When $\tilde{F}$ is splitted into m-bit blocks as

$$\tilde{F} = (\tilde{F}_1, \cdots, \tilde{F}_s),$$

it is easily shown that each $\tilde{F}_i$, $i = 1, \ldots, s$, depends only on βm bits of an n-bit input X: precisely we can write

$$\tilde{F}_i(X) = U_i\left(S_i\left([L_{i,i}|L_{i,i+1}|\cdots|L_{i,i+\beta-1}] \cdot X' + C_i'\right)\right) + D_i'$$

where S_i is an m-bit S-box in S layer, X' denotes the βm bits of X from the i-th m-bit block to $(i+\beta-1)$-th m-bit block, and C'_i and D'_i are the i-th m-bit block of C and $\tilde{M}^{-1} \cdot D$, respectively. In this way, we can view $\tilde{F}_i$ as an ASA structure based on a single m-bit S-box that takes as input βm bits and outputs m bits.

The first step of this phase is to fix $(\beta-1)m$ bits of inputs X' for each $\tilde{F}_i$ and then apply the affine equivalence algorithm of Lemma 1 to the resulting m-bit to m-bit ASA structure. Since the affine map A is invertible, the $m \times \beta m$ matrix

$$[L_{i,i}|L_{i,i+1}|\cdots|L_{i,i+\beta-1}]$$

has full row rank$(=m)$ over $\mathbb{F}_2$, and hence the column rank m. In order to find the positions of m linearly independent columns from this unknown matrix, we fix a set of m positions of X', and then evaluate $\tilde{F}_i$ for all the possible 2^m values on this set of positions with the other positions fixed as zero. If all the possible outputs of $\mathbb{F}_2^m$ are obtained from this evaluation, then the columns corresponding to these m positions would be linearly independent.

The probability that we choose m linearly independent columns from βm columns is $(1-\frac{1}{2})\cdot(1-\frac{1}{2^2})\cdots(1-\frac{1}{2^m}) > 0.288$ for the random full rank $m \times \beta m$ matrix. So, we would iterate the procedures to guess m positions of X' and check if all the possible outputs come out for about 5 times in average. It takes n^3 time to compute $\tilde{M}^{-1}$ and for each iteration, $nm2^m$ time to perform a matrix multiplication and $m2^m$ time to sort 2^m instances, with 2^m chosen plaintexts needed. Since five iterations would be held for each $1 \le i \le s$, it takes totally $n^3 + 5s(nm2^m + m2^m) = n^3 + 5(n^2+n)2^m$ steps with $5s2^m$ chosen plaintexts to find the positions of m linearly independent columns for all $1 \le i \le s$.

After this step, by fixing the other $(\beta-1)m$ positions of X' as zero, we obtain an invertible m-bit ASA structure. By applying the affine equivalence algorithm of Lemma 1 to this small construction which takes $m^3 2^{2m}$ time, we can recover the affine layers of $\tilde{F}_i$ for every $i = 1, \ldots, s$, and hence F. More precisely, after running the affine equivalence algorithms, we achieve U_i, C'_i, D'_i and the m linearly independent columns of $[L_{i,i}|L_{i,i+1}|\cdots|L_{i,i+\beta-1}]$. We recover the affine maps A and B from this information as follows. We first recover B multiplying $\tilde{M}$ to the affine map $U \cdot X + (D'_1, \cdots, D'_s)$ in time n^3, and compute B^{-1} in time n^3. Then the unknown $(\beta-1)m$ columns of $[L_{i,i}|L_{i,i+1}|\cdots|L_{i,i+\beta-1}]$ remain for each i. The j-th unknown column of this matrix is obtained by

$$S_i^{-1}(i\text{-th } m\text{-bit block of } (B^{-1} \cdot F(e_j))) + C'_i,$$

where e_j is the j-th coordinate vector in $\mathbb{F}_2^n$. To calculate all of them for $1 \le i \le s$, we need to compute $B^{-1} \cdot F(e_j)$ for all j, which takes $n \cdot (n^2)$ time with n chosen plaintexts. Now, we can obtain the whole matrix $[L_{i,i}|L_{i,i+1}|\cdots|L_{i,i+\beta-1}]$ for each i, and finally achieve A.

The overall work factor of the second phase is $4n^3 + 5(n^2+n)2^m + nm^2 2^{2m}$ with $s(5 \cdot 2^m + m)$ chosen plaintexts.

We can conclude the overall work factor in our attack including the first and second phases would be calculated as

$$\begin{aligned} sn[(\beta m+5)^2+5(\beta m)^2]+4n^3+5(n^2+n)2^m+nm^2 2^{2m} \\ \approx 6\beta^2 n^2 m+4n^3+5n^2 2^m+nm^2 2^{2m}, \end{aligned}$$

with about $s(2\beta m+5\cdot 2^m+m+10)$ chosen plaintexts.

Example 2. For $n=128$, $m=8$ and $\beta=3$, the time complexity of our attack would grow up to 2^{29}. For $n=256$, $m=8$ and $\beta=2$, the complexity would be less than 2^{31}. In these examples, the complexity of our attack algorithm is dominated by the term $nm^2 2^{2m}$.

3.3 Generalizations

In Sects. 3.1 and 3.2, we cryptanalyze the three-layer scheme ASA with specific input affine layers. We would provide an upper bound for the complexity of the attack algorithm for ASA with structured input affine layers.

Theorem 1. *Consider a three-layer scheme ASA, $F=B\circ S\circ A$ on n bits for which A is a structured affine mapping with respect to block length m and S is a concatenation of m-bit S-boxes. One can solve the specialized affine equivalence problem for F in time*

$$5\cdot\left(\frac{n}{m}\cdot\log_2\frac{n}{m}\right)\cdot n^3+5\cdot n^2\cdot 2^m+n\cdot m^2\cdot 2^{2m}$$

with $\frac{n}{m}(2n+5\cdot 2^m+m+10)$ chosen plaintexts.

Proof. The proof of theorem follows the attack scenario of Sects. 3.1 and 3.2. Since the attack procedure in the second phase is appliable to the general cases with no changes in time complexity, it suffices to show the following claim related to the first phase (with the same notations as in Sects. 3.1 and 3.2).

Claim. Let col_i be the column space obtained by picking plaintexts with no differentials except the i-th block in *Phase 1* (e.g. In our example in Sect. 3.1, $\mathsf{col}_i=\mathsf{col}(M_{i-\beta+1}|M_{i-\beta+2}|\cdots|M_i)$ for $1\le i\le s$). Given col_i for $1\le i\le s$, performing less than $s(\log_2 s+1)$ operations of intersections of subspaces in $\mathbb{F}_2^n$,[1] we can achieve bases for $\mathsf{col}(M_i)$ for $1\le i\le s$ over $\mathbb{F}_2$, respectively.

Proof of Claim (Sketch). Note that since L is invertible, every column of B_L is not a zero vector. The following algorithm terminates in $\log_2 s$ iterations and outputs $\mathsf{col}(M_i)$ for some single strip M_i.

– Let l be an index in $\{1,\cdots,s\}$. Set the initial values $v\leftarrow$ (the l-th column of B_L) and $\mathsf{col}\leftarrow\mathsf{col}_l$. We iterate the followings while $k>1$.

[1] Each operation of subspaces takes less than $5n^3$ steps by Lemma 4.

  - $k \leftarrow$ (hamming weight of v).
  - Let $\{i_1 < \cdots < i_k\}$ be the set of indices in which components of v are nonzeros.
  - For the i_1-th row and i_2-th row of B_L, find j such that the i_1-th component of the j-th column of B_L is different from the i_2-th component of the j-th column (such j exists since L is structured).
  - Set w as the j-th column of B_L.
    * If w has more than $\lfloor k/2 \rfloor$ nonzero overlapped components with v, then $v \leftarrow v + (v \wedge w)$ where "$\wedge$" indicates componentwise multiplication and compute $\mathsf{col} \leftarrow \mathsf{col} \cap \mathsf{col}_j^{\perp}$ where $\mathsf{col}_j^{\perp} \in \mathbb{F}_2^n$ is an orthogonal space of col_j.
    * Otherwise, set $v \leftarrow v \wedge w$ and compute $\mathsf{col} \leftarrow \mathsf{col} \cap \mathsf{col}_j$.
- Output v and col.

Remark 1. The algorithm outputs v whose components are all zeros except one. Suppose that the output $v \in \mathbb{F}_2^s$ has all zero entries except the i-th entry. Then, we can observe that the output col is equal to $\mathsf{col}(M_i)$. In other words, v indicates the index of the strip of which column space is obtained from the above algorithm.

Note that this algorithm does not guarantee to output distinct column spaces. So, to find distinct column spaces, we remove the indices i's from the initial $\{i_1, i_2, \cdots, i_k\}$, check if the set remains nonempty (if it is empty, then choose another l and repeat), and then replace the initial col with an intersection of col and the spaces $\mathsf{col}(M_i)^{\perp}$'s to run the algorithm again. Totally, we could output $\mathsf{col}(M_i)$ for $1 \le i \le s$ with $\log_2 s + (s-1)(\log_2 s + 1)$ operations of subspaces in $\mathbb{F}_2^n$. Though the above algorithm is not optimized for a particular A, it provides an approximate upper bound of complexity of finding $\mathsf{col}(M_i)$'s for the structured A with our strategies in general. □

4 Application to the White-Box AES Implementation

To see the background of the BCH implementation, let us take a glance at the historical aspects briefly. In the first white-box implementations presented by Chow et al. [6], the composition of a linear map and a nonlinear permutation with multiple S-boxes is used as an encoding. The linear map in their encoding contains a block diagonal matrix in which block provides a linear mixing bijection. However, the implementation is vulnerable to the Billet et al. attack [3]. Since then, Xiao and Lai proposed a white-box AES implementation with linear mappings as encodings [18]. They expected their implementation would resist the Billet et al. attack, using the linear encodings of block diagonal matrices whose block size is twice of the size of S-boxes. But the implementation was also broken by Mulder et al. attack [14] using linear equivalence algorithm in [4].

Recently, Baek et al. [1] showed that the substitution layers of the encodings in the previous constructions do not help to improve the security of the white-box implementations and the linear parts of the affine input encodings should not be split into the block diagonal matrices of small blocks to resist their attack toolbox. Hence, they constructed the special input encoding in which linear part can not be split, called *sparse unsplit encoding.* They presented their white-box AES implementation using the sparse unsplit encodings in [1], which was claimed to be secure against all known attacks including their attack toolbox.

However, the special structure of their sparse unsplit encodings threw new light on the cryptanalysis for us. We will explain our attack against the BCH implementation in this section. We can efficiently extract the round key of the implementation for all rounds except the first round, in 2^{33} time with 2^{14} chosen plaintexts for $n = 256$. This attack can also be applied for other parameters. The attack complexities for other parameters are presented in Table 2.

4.1 The BCH Implementation

The strategy of the BCH implementation is to obfuscate several parallel AES round functions at the same time using the special input encoding and to decompose the encoded round function into table lookups with small inputs so that their composition is equivalent to the encoded round function. Especially, the structured affine mapping with respect to the block length 8 was used as an input encoding in the BCH implementation.[2]

Let an input encoding $\widehat{\mathsf{A}}^{(r)}$ be a structured affine mapping on n bits with respect to block length 8 of the form in Eq. 1 for $\beta = 2$. The r-th encoded round function $F^{(r)}$ of AES-128 in the BCH implementation is of the form:

$$F^{(r)} = \widehat{\mathsf{B}}^{(r)} \circ \underbrace{(\hat{S}, \cdots, \hat{S})}_{\#\text{ of S-boxes}=s} \circ \oplus_{\underbrace{(K^{(r)}, \cdots, K^{(r)})}_{\#\text{ of round Key}=n/128}} \circ \widehat{\mathsf{A}}^{(r)},$$

where $\hat{S}$ is the S-box on 8 bits used in Rijndael, $K^{(r)}$ is the r-th round key of 128 bits in AES-128, and the output encoding $\widehat{\mathsf{B}}^{(r)}$ is an affine map defined as $\widehat{\mathsf{B}}^{(r)} = (\widehat{\mathsf{A}}^{(r+1)})^{-1} \circ (\mathsf{MC} \circ \mathsf{SR}, \cdots, \mathsf{MC} \circ \mathsf{SR})$ for $r < 10$, where MC and SR are the functions of $\mathsf{MixColumn}$ and $\mathsf{ShiftRow}$ steps in AES-128, respectively. Then, the encoded round function $F^{(r)}$ in the BCH implementation has ASA structure on n bits with $n = 8s$, where the S layer is a concatenation of s S-boxes on 8 bits and the input affine layer contains structured input affine mapping.

4.2 Cryptanalysis of the BCH Implementation

In our notations of Eq. (1), the input encoding of the BCH implementation is the case of $\beta = 2$ and $m = 8$. Hence, our cryptanalysis can be directly applied to the BCH implementation, setting $m = 8$. The encoded round function of the BCH Implementation is of the form in Sects. 3.1 and 3.2 for $\beta = 2$. For each round, we can solve the specialized affine equivalence problem for $F^{(r)}$ in

$$6\beta^2 n^2 m + 4n^3 + 5n^2 2^m + nm^2 2^{2m}$$

time with $s(2\beta m + 5 \cdot 2^m + m + 10)$ chosen plaintexts.

[2] In [1], they called the input encodings used in the BCH implementation as the *sparse unsplit* affine mapping.

We would regard $\widehat{\mathsf{B}}^{(r)}$ as B, and $\oplus_{(K^{(r)},\cdots,K^{(r)})} \circ \widehat{\mathsf{A}}^{(r)}$ as A, according to the notations in Sects. 3.1 and 3.2. For example, to find the image space of M_1, we would start with the plaintexts P_1, P_2, P_3 and P_4 such that P_1 and P_2 have the same values except the first 8-bit blocks, and P_3 and P_4 have same values except the second 8-bit blocks. From such plaintexts, we can find the column spaces as follows:

$$\mathsf{col}(M_1|M_s) = \left\{F(P_1) + F(P_2) \mid P_1, P_2 \in \{0,1\}^n \text{ with } P_1 + P_2 = (*, \mathbf{0}, \cdots, \mathbf{0}) \in \{0,1\}^{8\cdot s}\right\},$$

$$\mathsf{col}(M_1|M_2) = \left\{F(P_3) + F(P_4) \mid P_3, P_4 \in \{0,1\}^n \; with \, P_3 + P_4 = (\mathbf{0}, *, \mathbf{0}, \cdots, \mathbf{0}) \in \{0,1\}^{8\cdot s}\right\}$$

The column space $\mathsf{col}(M_1)$ is obtained by computing an intersection of $\mathsf{col}(M_1|M_s)$ and $\mathsf{col}(M_1|M_2)$. The work factor of the first phase in Sect. 3.1 is $sn[(2m+5)^2 + 5(2m)^2] \approx 2^{24}$ for $n = 256$.

In the second phase, for example, we know $\tilde{M}_1$ such that $M_1 = \tilde{M}_1 \cdot U_1$ for some (unknown) 8×8 matrix U_1. So, we have the function

$$\tilde{F}_1 = U_1 \circ \hat{S} \circ ((L_{1,1}|L_{1,2}) \cdot X' + C_1') + D_1',$$

where $\hat{S}$ is the 8-bit S-box in Rijndael, X' consists the first and second 8-bit blocks of an n-bit input X, and C_1' and D_1' are the first 8-bit blocks of C and $\tilde{M}^{-1} \cdot D$. To transform $\tilde{F}_1 : \mathbb{F}_2^{16} \to \mathbb{F}_2^8$ into an invertible map $\hat{F}_1$, we search for the set of eight indices $\{i_1, \cdots, i_8\}$ such that the output values of $\tilde{F}_1$ restricting j-th bits to be zeros for all $j \in \{1, \cdots, 16\} \backslash \{i_1, \cdots, i_8\}$ covers all 2^8 possible values. After then, applying Lemma 1 for $\hat{F}_1$ and $\hat{S}$, we can obtain U_1, C_1', D_1' and the eight columns of $[L_{1,1}|L_{1,2}]$. Each unknown column of $[L_{1,1}|L_{1,2}]$ can be recovered by computing $\hat{S}^{-1}$(the first 8 bits of $B^{-1} \cdot F(e_j))) + C_1'$ for $j \in \{1, \cdots, 16\} \backslash \{i_1, \cdots, i_8\}$. The overall complexity of the second phase is $4n^3 + 5(n^2+n)2^m + nm^2 2^{2m} \lesssim 2^{31}$ for $n = 256$.

Hence, we can recover a pair A and B, a solution for the specialized affine equivalence problem in 2^{31} time for $n = 256$.

Extracting the Round Keys. Now, our goal is to extract the round key bits except for the first round. Note that it suffices to have the adjacent two round keys to extract the full 128-bit AES key.

Following the above strategies, we have possibly many candidates of $\widehat{\mathsf{B}}^{(r)}$ and $\oplus_{(K^{(r+1)},\cdots,K^{(r+1)})} \circ \widehat{\mathsf{A}}^{(r+1)}$ on consecutive rounds. However, just one representative of the solutions, say $B^{(r)}$ and $A^{(r+1)}$, would be used to recover the exact $\widehat{\mathsf{B}}^{(r)}$ and $\oplus_{(K^{(r+1)},\cdots,K^{(r+1)})} \circ \widehat{\mathsf{A}}^{(r+1)}$ and extract the $(r+1)$-th round key bits, with the set of self-equivalences of $\hat{S}$.

We know that the exact pair of $\oplus_{(K^{(r+1)},\cdots,K^{(r+1)})} \circ \widehat{\mathsf{A}}^{(r+1)}$ and $\widehat{\mathsf{B}}^{(r)}$ differs from the obtained pair $A^{(r+1)}$ and $B^{(r)}$ by the $2s$ pairs of affine self-equivalences of the S-box $\hat{S}$. Recall that

$$\widehat{\mathsf{A}}^{(r+1)} \circ \widehat{\mathsf{B}}^{(r)} = (\mathsf{MC} \circ \mathsf{SR}, \cdots, \mathsf{MC} \circ \mathsf{SR}).$$

Hence, to find the exact pair of $\oplus_{(K^{(r+1)},\cdots,K^{(r+1)})} \circ \widehat{\mathsf{A}}^{(r+1)}$ and $\widehat{\mathsf{B}}^{(r)}$, it suffices to find the set of self-equivalences of $\hat{S}$,

$$\{(a_1, b_1), \cdots, (a_s, b_s), (a_1', b_1') \cdots, (a_s', b_s')\}$$

such that

$$(\ell_{a_1}, \cdots, \ell_{a_s}) \circ L^{(r+1)} \circ M^{(r)} \circ (\ell_{b'_1}, \cdots, \ell_{b'_s}) = (\mathsf{MC} \circ \mathsf{SR}, \cdots, \mathsf{MC} \circ \mathsf{SR}),$$

where ℓ_{a_i}, ℓ_{b_j} are the linear part of the small affine maps a_i and b_j on 8 bits, respectively. So, we do the followings.

- Searching for all self-equivalences, we would find self-equivalences (a_1, b_1) and (a'_1, b'_1) of $\hat{S}$ such that
$$\ell_{a_1} \cdot [(1,1)\text{-th block of } L^{(r+1)} \cdot M^{(r)}] \cdot \ell_{b'_1}$$
is equal to the corresponding $(1,1)$-th block of the matrix $(\mathsf{MC} \circ \mathsf{SR}, \cdots, \mathsf{MC} \circ \mathsf{SR})$.
- If we find the right pairs (a_1, b_1) and (a'_1, b'_1), then fix b'_1 and then search for all self-equivalences to find (a_j, b_j) such that
$$\ell_{a_j} \cdot [(j,1)\text{-th block of } L^{(r+1)} \cdot M^{(r)}] \cdot \ell_{b'_1}$$
is equal to the corresponding $(j,1)$-th block of the matrix $(\mathsf{MC} \circ \mathsf{SR}, \cdots, \mathsf{MC} \circ \mathsf{SR})$ for all $1 \le j \le s$.
- Samely, fix a_1 and then search for all self-equivalences of $\hat{S}$ to find (a'_j, b'_j) such that
$$\ell_{a_1} \cdot [(1,j)\text{-th block of } L^{(r+1)} \cdot M^{(r)}] \cdot \ell_{b'_j}$$
is equal to the corresponding $(1,j)$-th block of the matrix $(\mathsf{MC} \circ \mathsf{SR}, \cdots, \mathsf{MC} \circ \mathsf{SR})$ for all $1 \le j \le s$.
- Now we have the set of $\{(a_1, b_1), \cdots, (a_s, b_s), (a'_1, b'_1) \cdots, (a'_s, b'_s)\}$ so that we can obtain
$$(a_1, \cdots, a_s) \circ A^{(r+1)} \circ B^{(r)} \circ (b'_1, \cdots, b'_s) = \oplus_{(K^{(r+1)}, \cdots, K^{(r+1)})} \circ \widehat{\mathsf{A}}^{(r+1)} \circ \widehat{\mathsf{B}}^{(r)}.$$

Since the number of self-equivalences of $\hat{S}$ is about 2^{11} by Lemma 3, the work factor to find the exact pair of $\oplus_{(K^{(r+1)}, \cdots, K^{(r+1)})} \circ \widehat{\mathsf{A}}^{(r+1)}$ and $\widehat{\mathsf{B}}^{(r)}$ is $[(2^{11})^2 + 2 \cdot (s-1) \cdot 2^{11}] \cdot (2 \cdot m^3) + 2 \cdot n^3 \approx 2^{32}$ for $n = 256$.

Now, we know the exact affine maps $\oplus_{(K^{(r+1)}, \cdots, K^{(r+1)})} \circ \widehat{\mathsf{A}}^{(r+1)}$ and $\widehat{\mathsf{B}}^{(r)}$. We can achieve the round key bits $K^{(r+1)}$ from

$$(\oplus_{(K^{(r+1)}, \cdots, K^{(r+1)})} \circ \widehat{\mathsf{A}}^{(r+1)}) \circ \widehat{\mathsf{B}}^{(r)} = \oplus_{(K^{(r+1)}, \cdots, K^{(r+1)})} \circ (\mathsf{MC} \circ \mathsf{SR}, \cdots, \mathsf{MC} \circ \mathsf{SR}),$$

in time complexity n^2. In fact, $(K^{(r+1)}, \cdots, K^{(r+1)})$ is the sum of the constant of $\widehat{\mathsf{A}}^{(r+1)}$ and $L^{(r+1)} \times$ (the constant of $\widehat{\mathsf{B}}^{(r)}$).

Thus, the total work factor of our attack for the BCH implementation to extract the round key is less than 2^{33} for $n = 256$. The complexity of our attack is stable for other parameters as in Table 2, since it mainly depends on the input size of S-boxes.

5 Conclusion

In this paper, we suggested an optimized algorithm to solve the affine equivalence problem in the case that the middle S layer is a concatenation of S-boxes and the input affine layer is structured. For the three-layer scheme $F = B \circ S \circ A$ satisfying our problem setting, one can find the secret affine layers via oracle queries to F (as black boxes) with our algorithm in low complexity. Our algorithm is more efficient than previous algorithms such as the affine equivalence algorithm [4] and SAEA [1].

The structured affine map could induce an efficient white-box implementation. In the BCH implementation [1], the structured affine mapping was used as an input encoding to resist known attacks. Baek et al. expected that their implementation is secure against a cryptanalysis using SAEA. In this paper, we showed that the overall work factor of SAEA can be significantly reduced. As a result, our cryptanalysis on the BCH implementation efficiently extracted the round key with low complexity, 2^{32}, 2^{33}, and 2^{34} for $n = 128, 256$, and 384, respectively.

Acknowledgements. This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-TB1403-03. The authors would like to thank the anonymous reviewers of SAC 2016 for their helpful comments.

References

1. Baek, C.H., Cheon, J.H., Hong, H.: White-box AES implementation revisited. J. Commun. Netw. **18**(3), 273–287 (2016)
2. Biham, E., Anderson, R., Knudsen, L.: Serpent: a new block cipher proposal. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 222–238. Springer, Heidelberg (1998). doi:10.1007/3-540-69710-1_15
3. Billet, O., Gilbert, H., Ech-Chatbi, C.: Cryptanalysis of a white box AES implementation. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 227–240. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30564-4_16
4. Biryukov, A., De Cannière, C., Braeken, A., Preneel, B.: A toolbox for cryptanalysis: linear and affine equivalence algorithms. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 33–50. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_3
5. Chow, S., Eisen, P., Johnson, H., van Oorschot, P.C.: A white-box DES implementation for DRM applications. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 1–15. Springer, Heidelberg (2003). doi:10.1007/978-3-540-44993-5_1
6. Chow, S., Eisen, P., Johnson, H., Van Oorschot, P.C.: White-box cryptography and an AES implementation. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 250–270. Springer, Heidelberg (2003). doi:10.1007/3-540-36492-7_17
7. Daemen, J., Rijmen, V.: AES Proposal: Rijndael (1999)
8. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: the Even-Mansour scheme revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 336–354. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_21

9. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. J. Cryptol. **10**(3), 151–161 (1997)
10. Karroumi, M.: Protecting white-box AES with dual ciphers. In: Rhee, K.-H., Nyang, D.H. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 278–291. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24209-0_19
11. Leander, G., Poschmann, A.: On the classification of 4 bit S-boxes. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 159–176. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73074-3_13
12. Liu, F., Ji, W., Hu, L., Ding, J., Lv, S., Pyshkin, A., Weinmann, R.-P.: Analysis of the SMS4 block cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 158–170. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73458-1_13
13. Michiels, W., Gorissen, P., Hollmann, H.D.L.: Cryptanalysis of a generic class of white-box implementations. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 414–428. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04159-4_27
14. De Mulder, Y., Roelse, P., Preneel, B.: Cryptanalysis of the Xiao – Lai white-box AES implementation. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 34–49. Springer, Heidelberg (2013). doi:10.1007/978-3-642-35999-6_3
15. Özbudak, F., Sınak, A., Yayla, O.: On verification of restricted extended affine equivalence of vectorial boolean functions. In: Koç, Ç.K., Mesnager, S., Savaş, E. (eds.) WAIFI 2014. LNCS, vol. 9061, pp. 137–154. Springer, Cham (2015). doi:10.1007/978-3-319-16277-5_8
16. Saarinen, M.-J.O.: Cryptographic analysis of all 4 × 4-bit S-boxes. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 118–133. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28496-0_7
17. Wan, Z.: Geometry of Classical Groups over Finite Fields, 2nd edn. Science Press, Beijing (2006)
18. Xiao, Y., Lai, X.: A secure implementation of white-box AES. In: Computer Science and its Applications - CSA 2009, pp. 1–6. IEEE (2009)