An Efficient Structural Analysis of SAS and

its Application to White-Box Cryptography

Hyoungshin Yim

*Dept. of Financial information security, Kookmin University*

Seoul, South Korea

kuunh2@kookmin.ac.kr

Ju-Sung Kang

*Dept. of Information Security, Cryptology, and Mathematics/*

*Financial information security, Kookmin University*

Seoul, South Korea

jskang @ kookmin.ac.kr

Yongjin Yeom

*Dept. of Information Security, Cryptology, and Mathematics/*

*Financial information security, Kookmin University*

Seoul, South Korea

salt @ kookmin.ac.kr

*Abstract*—Structural analysis is the study of finding component functions for a given function. In this paper, we proceed with structural analysis of structures consisting of the S (nonlinear Substitution) layer and the A (Affine or linear) layer. Our main interest is the structure with different substitution layers and large input/output sizes. The purpose of our structural analysis is to find the functionally equivalent oracle and its component functions for a given encryption oracle . As a result, we can construct the decryption oracle explicitly and break the one-wayness of the building blocks used in a White-box implementation. Our attack consists of two steps: S layer recovery using multiset properties and A layer recovery using differential properties. We present the attack algorithm for each step and estimate the time complexity. Finally, we discuss the applicability of structural analysis in a White-box Cryptography environment.

Keywords— cryptography; structural analysis; SAS strcuture; white-box; security

# Introduction

Structural analysis is the study of revealing internal components of a function provided by the structure and unknown component functions. In other words, A given cryptographic oracle in a block-box model can be viewed only in terms of input/output values without knowledge of internal functions. The structural analysis is informally defined as:

* *Structural analysis*: Given an encryption oracle and its structure, that is, the assumption that the structure of the oracle is known as illustrated in Fig. 1, we define the structural analysis of by the analysis to find an equivalent oracle by revealing all internal components of explicitly.

When a function is used as a building block of a cryptographic algorithm, it may include subfunctions that are key-dependent or protected secret components during the cipher operation. For more than two decades, cryptographic study on the structural analysis has been developed particularly on the layered structure which alternates S(nonlinear Substitution) and A(Affine or linear) layers such as SAS, ASASA, and SASAS. These structures are shown in Fig. 1. S layer consists of nonlinear S-boxes in parallel and A layer represents a bitwise linear transformation. We focus on SAS structure with different size S-boxes in the first and the last S layers.

White-box cryptography (WBC) is a method of mixing and hiding encryption key with other components in software implementations of a cryptographic algorithm. That is, it is a technique to perform a cryptographic algorithm without revealing encryption keys or sensitive security parameters. The WBC offers one-wayness properties and key hiding techniques. One-wayness means the unidirectional operation of in that it is infeasible to execute a decryption oracle given an encryption oracle . The key hiding technique enables WBC implementation to cloak the key during the encryption process by merging key-dependent operations in the lookup tables. SAS structure is widely used as one of the main building blocks in white-box cryptography.

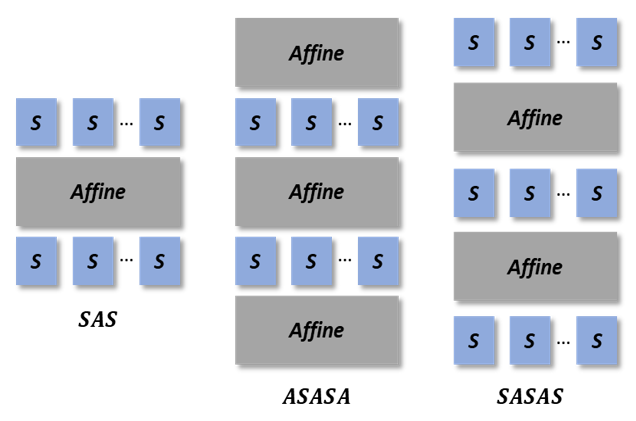


Fig. 1. Examples of substitution/affine structures

In this paper, we suggest an efficient inverting method of SAS structure, which consists of alternating substitution and affine structure. Particularly, we analyze the generalized structure where the sizes of the components in two substitution layers are different (e.g. 5 bits for and 4 bits for ). Our analysis works for SAS with large input/output size. For a WBC construction with 60-bit oracles, we can construct an attack on the main building blocks by inverting oracle with structural analysis.

# Related Work

In this section, we present previous research for structural analysis and explain several properties of *Multisets.*

## Researches on alternating Substitution/affine structures

Structural analysis consisting of substitution/affine functions has been studied for two decades, as listed in Table Ⅰ.

In 2001, Biryukov and Shamir [1] proposed how to efficiently attack the substitution/affine layer in a SASAS structure with the same size of S-boxes for all substitution layers. In 2003, Biryukov et al. [2] has presented a detailed analysis method for the affine layer or linear layer. An improved method for finding an affine equivalent function was recently proposed in [8]. In 2014, analysis of ASASA as black-box, white-box, and public-key cryptography was conducted in [3]. Research on this structure has been ongoing [4][6] and even SASASASAS structure was conducted in [5].

TABLE I. Papers Related To The Subtitution/Affine Structure

|  |  |  |
| --- | --- | --- |
| **Year** | **Topic** | **Authors** |
| 2001 | Structural cryptanalysis of SASAS [1] | A. Biryukov,  A. Shamir, et al. |
| 2003 | Affine Equivalence Algorithms [2] | A. Biryukov,  B. Preneel, et al. |
| 2014 | Cryptographic Schemes Based on the ASASA [3] | A. Biryukov,  C. Bouillauet, et al. |
| 2015 | Structural cryptanalysis of ASASA [4] | I. Dinur,  O. Dunkelman, et al. |
| 2015 | Decomposition attack on SASASASAS [5] | A. Biryukov,  D. Khovratovich |
| 2015 | Key-Recovery Attack on the ASASA [6] | H. Gilbert,  J. Plut, et al. |
| 2016 | Analytic Tools for White-box Cryptography [7] | CH. Baek |
| 2018 | An Improved Affine Equivalence Algorithm [8] | I. Dinur |

## Multiset properties

We introduce *Multiset* properties[1] that allow characterizing the intermediate values of an encryption structure, even if none of the actual functions is known.

**GOAL** (Efficient structural analysis of )

Suppose that an encryption oracle is given with structure and different substitution layers. That is, we know the construction of with its unknown components and we can compute for any input. The goal of the structural analysis is to construct an equivalent oracle explicitly by determining all its components so that we can compute for any as well.

#### Multiset allows for multiple instances for each of its elements. Properties of a Multiset are shown below :

* *Property P* (Permutation): all element must appear exactly once.
* *Property E* (Even): each element included an even number of times.
* *Property D* (Dual): it is either *property P* or *property E*.
* *Property B* (Balanced): the *XOR* of all elements is the zero vector.

#### Multisets may have specific properties for substitution/affine structure. For simplicity, consider a 60-bit oracle with . We define Multisets as follows: Let input for, where is a Galois field of 32 elements . The input is expressed as input for each of the S-boxes in the first substitution layer. A Multiset defined for the input as for , where means a Multiset corresponding to each S-box in the first substitution layer, and when observing this property, look at (). When the Multiset is applied to the structure in this paper, each layer meets the following conditions, as shown in Fig. 2.

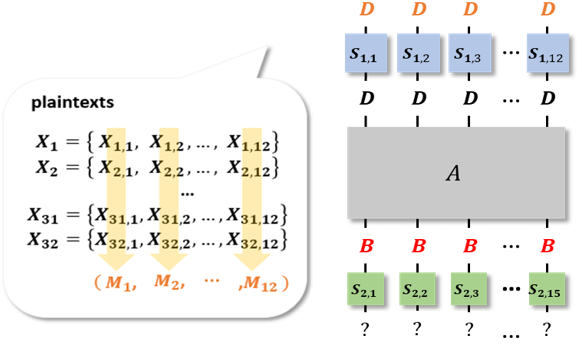


Fig. 2. *Multiset* properties in structure

#### Through Fig. 2, if the Multiset for the input satisfies (D, D, ..., D), the input Multiset of the second substitution layer satisfies (B, B, ..., B). Note that the input Multiset of the second substitution layer becomes a 15-dimensional vector as () because the S-boxes have a 4-bit input. This feature contributes to the recovery of the second S-boxes. The following section details how to attack structures.

# Attack of SAS Structure

This section describes a detailed attack for analysis on structure, which has substitution layers of different sizes. Therefore, the goal of structural analysis in this paper is as follows:

As with the above goal, we obtain encryption oracle and decryption oracle , which are functionally equivalent to encryption oracle . Oracle and the obtained oracle depict in Fig. 3. For any input , and , then the same output . The figure shows that the input and output are identical, but the subfunctions constructed inside the oracle are different. Note that, is not uniquely determined. We successfully find one of the several ’s. The attack method proposed in this paper consists of two steps. The first step is to convert into, a combination of linear and affine, to find . The second step finds in the structure obtained in step 1. Finally, oracle , which is functionally equivalent to , can be found. The attack proceeds in two steps described below.

Step1) **Algorithm1** (

Find such that

Step2) **Algorithm2** ):

Find such that

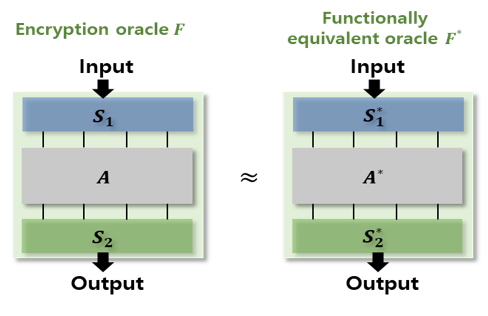


Fig. 3. Encryption oracle and functionally equivalent oracle

## Components in structural analysis of oracle

Each function and component defined in this paper shown below:

* encryption oracle scheme.
* invertible nonlinear function.

|  |  |
| --- | --- |
|  | (1) |

* invertible nonlinear function.

|  |  |
| --- | --- |
|  | (2) |

* invertible affine function.

|  |  |
| --- | --- |
|  | (3) |

## Components in structural analysis of target oracle

is the target oracle that we have to construct. , , and are the internal functions. The components obtained at each step shown below:

* encryption target oracle.
* invertible linear function.

|  |  |
| --- | --- |
|  | (4) |

* invertible linear function.

|  |  |
| --- | --- |
|  | (5) |

* invertible nonlinear function.

|  |  |
| --- | --- |
|  | (6) |

* invertible nonlinear function.

|  |  |
| --- | --- |
|  | (7) |

* invertible affine function.

|  |  |
| --- | --- |
|  | (8) |

The component functions , and are not uniquely determined since invertible linear functions in and can be arbitrarily chosen. We describe the attack against structures using fixed parameters (60-bit oracle with 4-bit/5-bit S-boxes).

## Recovering S-box layer from

**Algorithm 1** recovers the second S-box layer from encryption oracle . In other words, “”, it recovers from and reconstructs it into an structure. The process removes the last S layer as: .

|  |  |
| --- | --- |
| **Algorithm 1.** Recovering S-boxes layer algorithm | |
| **Input:** encryption oracle  **Output:** nonlinear function such that | |
| 1. | **For** **each** **in** **do** |
| 2. | Choose input values that satisfies properties (*D, D, ..., D*) |
| 3. | Obtain the corresponding output values of |
| 4. | Indexing the input values of by output values |
| 5. | Establish a linear equation using *Multiset* properties (*B, B, ..., B*) |
| 6. | Construct a system of linear equations |
| 7. | Use Gaussian elimination to obtain a non-zero solution |
| 8. | || || … || ) |
| 9. | || || … || ) // *inverse of* () |

This attack carries out in order of each S-boxes on the second layer of substitution. Suppose that we attack the first S-box in the second substitution layer. We choose plaintexts that satisfy the properties (*D, D, ..., D*) and corresponding ciphertexts. As explained in Section Ⅱ, if the plaintext satisfies (*D, D, ..., D*) properties, then the *Multisets* of inputs have (*B, B, ..., B*) properties. We proceed with the attack using the sum of all elements, which are balanced properties, zero. First, index the input corresponding to all variables that can come to the S-box output. We set to the input of the S-box such that , shown in Table Ⅱ.

TABLE II. Indexing the Input Values Z of

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Z*** |  |  | … |  |
| ***Y*** | 0 | 1 | … | 15 |

Since the S-box in the second layer is a nonlinear function from 4-bits onto 4-bits, indices are used. Next, a linear equation creates using the corresponding indexes of elements constructed from the *Multiset* of the obtained ciphertexts. That is, when the *Multiset* of the output for one S-box is , the equation is as follows: . Let's take a look at the detailed explanation with the toy example. Let us set ciphertexts to . We obtain a *Multiset* for the first S-box when the ciphertexts we obtain are equal to the following , , …, . As a result, and produces a linear equation that satisfies balanced properties.

|  |  |
| --- | --- |
|  | (9) |

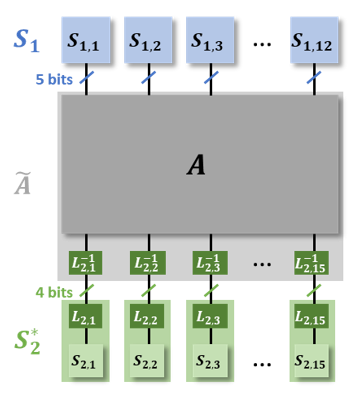


Fig. 4. Modified structure in the S-box recovery phase

To find all solutions, we need 16 equations like (7). Thus, the process performs 16 times to obtain an equation. Then, given solution vector , 16 linear equations are expressed as (: 16×16 matrix over ). However, matrix has a kernel of dimension 5, and rank is 11. This is because it is possible to determine the 5 free variables to select the linear transformation . That is, it cannot have a full rank. Consequently, it has multiple solutions, as suggested in [1]. After all, we obtain 11 linearly independent equations after applying Gaussian elimination, which cannot determine a single solution. Therefore, as shown in Fig. 4, the obtained S-box in possibly contains a linear function. When attacking using Algorithm 1, we consider the S-box and the affine function which contain parts of linear functions canceled out as in Fig. 4. The following section describes the process of recovering the affine layer from the obtained structure.

## Recovering affine layer from

**Algorithm 2** recovers the affine layer. In other words, “” it recovers from , and we can obtain an . The process is formulated as follows: .

|  |  |
| --- | --- |
| **Algorithm 2.** Recovering affine layer algorithm | |
| **Input:** oracle  **Output:** affine function such that | |
| 1. | is the set of possible input differences at |
| 2. | **For** **each** **in** **do** |
| 3. | *// is the set of 5-bit input differences* |
| 4. | *// is the set of 60-bit output differences* |
| 6. | **Repeat** |
| 7. | *// Select one randomly from set* |
| 8. | *// is the i-th difference of* |
| 9. | **if** *dim*() *dim*() **then** |
| 10. |  |
| 11. | //*Update* *the difference set* |
| 12. | **end if** |
| 13. | **until** *dim*() //*Calculate dimension of the subspace  generated by* |
| 14. | Find the basis of the |
| 15. | Obtain affine layer , where each basis is a column vector |
| 16. | || || … ||) |

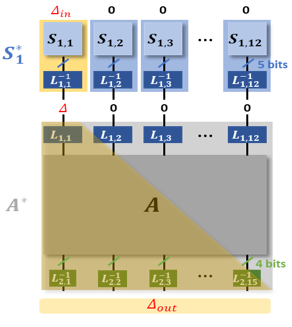


Fig. 5. Modified structure in the affine recovery phase

Attacks proceed in order from the first S-box. Suppose that we attack the first S-box. The key idea of this attack is to calculate the difference ( in Fig. 5) by selecting two plaintexts from the corresponding S-box. For the input and of the attack location S-box, the difference should not be zero (). The input difference of the other location, S-box, should be zero. Set the difference between each S-box to one input difference .

As depicted in Fig. 5, only the difference set corresponding to the first S-box puts an input that affects the output difference . The reason for this setting is that the difference characteristics of the first S-box corresponding to the location of the attack affect the output. As such, outputs obtained using the difference condition are attacked by using the basis transformation in the output difference values from . To recover the affine layer , the important point is that the elements of set must satisfy their independence from each other. The verification method confirms the changes in the dimension of set , which adds the calculated difference with . If dimension *dim*() is greater than dimension *dim*(), then the difference values in set and are independent of each other. Therefore, the calculated differences under that condition are stored in , a set of input differences. Through the above process, we construct an independent five-difference set . If the output dimension becomes 5, the dimension of the final output dimension becomes 5 or less. Therefore, if we find a case where the confirm the independent because only the case where the dimension of the output values is five.

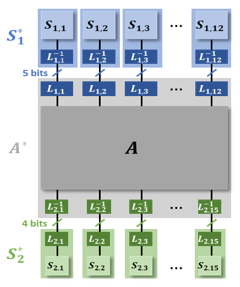


Fig. 6. An function functionally equivalent to encryption oracle

The affine layer is the basis transformation of the sets obtained, and the process shows below. Although we do not know what the basis is for the difference output values of the original , by adding a linear function, we assume that five linearly independent output differences of form the standard basis. The standard basis expresses as and affine layer performs, the following equation is satisfied:

|  |  |
| --- | --- |
|  | (10) |

The is the basis for ciphertexts. Eventually, the affine layer is represented by a matrix of , which is a blockwise vector. Then, matrix is composed of a matrix.

|  |  |
| --- | --- |
|  | (11) |

If we will use these features, the part of the affine layer recovered. Like the above process, recover the affine layer by performing the same procedure for other S-box.

Through Algorithm1 and Algorithm2, we were able to obtain, , and from the structure. The oracle consisting of the obtained functions is functionally equivalent to the encryption oracle . After all, since we know and , we can obtain each function’s inverse function and generate a decryption oracle . Fig. 6 shows an function that recovered the structure.

# Complexity of Attack

In this section, we present the complexity based on the algorithms presented in Section Ⅲ.

## Complexity of recovering S-box layer

The main operation in the S-box recovery phase is Gaussian elimination. First, since a Gaussian elimination operation performs for each S-box, a total of times achieve. To be more specific, when we set up the linear equations in each S-box, it is represented as , and becomes a matrix. Using Gaussian elimination in this matrix results in the time complexity of . The reason is in the process of proceeding with the Gaussian elimination method. The first phase in Gaussian elimination is equations with variables, so pivots are required, with a complexity of The next phase is to subtract multiple rows from each other’s rows for each pivot, for which the complexity is . Thus, the total runtime is . However, since we have a total of -second layer S-boxes, it becomes . By expressing it in general, it describes as follows:

|  |  |
| --- | --- |
|  | (12) |

means the number of S-boxes in the second substitution layer, and means the input/output size of each S-box. The calculated complexity is the same as the complexity presented in [1] paper.

## Complexity of recovering Affine layer

Like the S-box recovery phase, the main operation of the affine layer recovery phase is Gaussian elimination. Five 60-bit outputs of the structure form a matrix. It was using Gaussian elimination in this matrix that results in the time complexity of . First, since there are equations, it needs pivots, and the time complexity for this is . Next, the multiplication and subtraction operations perform using each row for each pivot calculation. The time complexity for this becomes . But, since this process has to be performed for each S-box, the time complexity is equal to . By expressing it in general, it describes as follows:

|  |  |
| --- | --- |
|  | (13) |

means the number of S-boxes in the first substitution layer, and means the input/output size of each S-box. , which means the input/output size of the entire oracle. As a result, time complexity of S-box recovery phase is and time complexity of affine recovery phase is . Thus, the attack on the entire structure is approximately, and the phase to recover affine layer is more time complexity than the S-box recovery phase.

# Application of SAS Structure in White-Box Cryptography

This section discusses how structural analysis, introduced in Section Ⅲ, applies to the WBC. structural analysis can generate a functionally equivalent decryption oracle when given an encryption oracle. This does not satisfy the one-wayness of the WBC. However, research on the WBC with SAS structure is still in progress. In 2019, Shi presented the WBC scheme with large block SAS structure [9]. Shi’s WB model has a Feistel structure and a SAS structure consisting of TBOX. The detailed structure for this Shi’s WB model summarizes in TABLE Ⅲ.

TABLE III. Structure Form of Shi’S Wb Model

|  |  |
| --- | --- |
| **Shi’s WB model** | |
| **Structures** | Feistel structure, SAS structure |
| **Total length** | 120 bits (60bits, 60bits) |
| **Round** | 16 |
| **S-box layers** | , |
| **Affine layer** |  |
| **Security claimed level** |  |

As in Table Ⅲ, we can confirm that it has the same class as the structure presented in this paper. In addition, substitution/affine is composed of a single T-box to provide oracle characteristics. The Feistel structure of Shi’s WBC, also exposes the input/output values of the TBOX since the input/output values expose by round. Furthermore, the internal function is an undisclosed in the white-box environment. A detailed structure is shown in Fig. 7.

The security claimed in Shi’s [9] base on components of different widths and an appropriate number of rounds. This paper argues that the time complexity for the attack is for 2.5 rounds. Such a security claim is based on the followings:

* The sizes of S-boxes in two layers are different 4 bits and 5 bits, respectively.
* (4, 5) are coprime so that it can obtain a relatively large least common multiplier for defeating the *Multiset* attack.

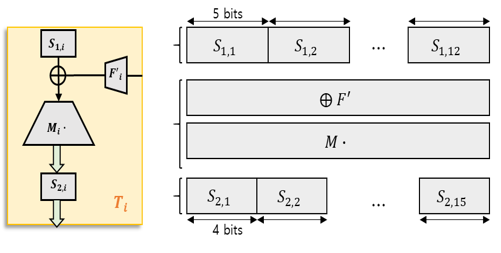


Fig. 7. SAS structure in Shi’s WB model

However, in this paper, we showed successful structural analysis in the SAS structures with the same class. In Section Ⅳ, we calculate the time complexity for this attack, which is much lower than the complexity claimed by Shi. We implemented the attack method proposed in this paper and the execution time measured within 1 second. Therefore, our result shows that Shi’s WBC model is suspected to be vulnerable.

# Conclusion

This paper proposes an efficient structural analysis of with large input/output sizes and different S layers. The purpose of structural analysis is to find a functionally equivalent decryption oracle for a given encryption oracle . We analyze the time complexity for the proposed attack algorithm and prove that it has an extremely low level of security. Furthermore, we also experimentally confirm that the attack is possible within one second with our implementation. Therefore, when WBC uses as a building block for one-wayness properties, our attack successfully breaks WBC within a few seconds depending on the number of look-up tables. As a future work, the structural analysis presented in this paper could break one-wayness of WBC scheme with the SAS structure.

##### Acknowledgment

This work has supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021M1A2A2043893).

##### References

1. A. Biryukov, A. Shamir, “Structural Cryptanalysis of SASAS,” Advances in Cryptology - EUROCRYPT 2001, LNCS 2045, Springer-Verlag, pp. 394–405, 2001.
2. A. Biryukov, C.D. Cannière, A. Braeken, and B. Preneel, “A toolbox for cryptanalysis: linear and affine equivalence algorithms,” Advances in Cryptology - EUROCRYPT 2003, LNCS 1267, Springer-Verlag, pp. 33–50, 2003.
3. A. Biryukov, C. Bouillaguet, and D. Khovratovich, “Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract),” ASIACRYPT 2014, Part I, LNCS 8873, pp. 63–84, Dec. 7–11, 2014.
4. I. Dinur, O. Dunkelman, T. Kranz, and G. Leander, “Decomposing the ASASA Block Cipher Construction,” Cryptology ePrint Archive, Report 2015/507, 2015.
5. A. Biryukov and D. Khovratovich, “Decomposition attack on SASASASAS,” Cryptology ePrint Archive, Report 2015/646, 2015.
6. H. Gilbert, Jérôme Plût, and J. Treger, “Key-Recovery Attack on the ASASA Cryptosystem with Expanding S-boxes,” Advances in Cryptology – CRYPTO 2015, Springer, pp. 475-490, 2015.
7. Chung Hun Baek, “Analytic Tools for White-box and Lattice Cryptography,” Ph. D. Thesis, Department of Mathematical Sciences, Seoul National University, 2016.
8. I. Dinur, “An Improved Affine Equivalence Algorithm for Random Permutation,” Advanceds in Cryptology –EUROCRYPT 2018, Springer, pp. 413-442, 2018.
9. Y. Shi, W. Wei, H. Fan, M.H. Au, X. Luo, “A Light-Weight White-Box Encryption Scheme for Securing Distributed Embedded Devices,” IEEE Transaction on Computers, pp. 1411-1427, 2019.