SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Cryptanalysis of the Obfuscated Round Boundary Technique for Whitebox Cryptography

Yongjin YEOM[1*], Dong-Chan KIM[1], Chung Hun BAEK[2] & Junbum SHIN[2]

[1] *Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul* 02707*, Republic of Korea;*
[2] *Samsung Research, Samsung Electronics, Seoul* 06765*, Republic of Korea*

Appendix A Xu et al.'s Whitebox Implementations

Appendix A.1 SLT cipher and AES

Consider a block cipher $E : \{0,1\}^n \to \{0,1\}^n$ composed of round functions f_i of the form

$$E = f_N \circ f_{N-1} \circ \cdots \circ f_1,$$

where f_i is called the round function of E $(1 \leqslant i \leqslant N)$. The round function $f_i : \{0,1\}^n \to \{0,1\}^n$ of an SLT cipher is divided into two layers: the substitution layer and linear transformation layer. The substitution layer S is a keyed nonlinear transform that uses S boxes $S_1, S_2, \ldots, S_k$ in parallel. For the sake of simplicity, we only consider the case of $n = 128$ and $k = 16$. Then S box S_j $(1 \leqslant j \leqslant 16)$ is defined as a byte-wise nonlinear bijection. The transform $S : X \mapsto Y$ is expressed as $X = (x_1, \ldots . x_{16}) \mapsto Y = (y_1, \ldots, y_{16})$ or

$$X = x_1 \| x_2 \| \cdots \| x_{16} \mapsto Y = S_1(x_1 \oplus rk_1) \| S_2(x_2 \oplus rk_2) \| \cdots \| S_{16}(x_{16} \oplus rk_{16}),$$

where $rk_1, \ldots, rk_{16}$ are round keys and $\|$ indicates the concatenation. The linear transformation layer $L : \{0,1\}^n \to \{0,1\}^n$ uses a $128 \to 128$ matrix M over $GF(2)$. The transform $L : Y \mapsto Z$ is expressed as

$$Z = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{16} \end{pmatrix} = M\,Y = \begin{pmatrix} M_1 & M_2 & \cdots & M_{16} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{16} \end{pmatrix} = y_1 M_1 \oplus y_2 M_2 \oplus \cdots \oplus y_{16} M_{16},$$

where M_j is the j-th column of the matrix M $(1 \leqslant j \leqslant 16)$.

The standard block cipher AES [1] is composed of 10 round functions, which are divided into 4 subfunctions: AddRoundKey, SubBytes, ShiftRows, and MixColumns. AES can be considered as an SLT cipher, the round function of which has the form

$$\underbrace{\textsc{MixColumns} \circ \textsc{ShiftRows}}_{\text{linear layer } L} \circ \underbrace{\textsc{SubBytes} \circ \textsc{AddRoundKey}}_{\text{substitution layer } S}.$$

Appendix A.2 Lookup tables for whitebox SLT cipher

For a fixed encryption key, each round function of an SLT cipher can be computed using 8-bit to 128-bit tables and exclusive-or operations on 128-bit data. In fact, with the round key $(rk_1, \ldots, rk_{16})$, the round function $R = L \circ S : \{0,1\}^{128} \to \{0,1\}^{128}$ maps $(x_1, \ldots, x_{16})$ to $(y_1, \ldots, y_{16})$ by

$$\begin{aligned} \begin{pmatrix} y_1 \\ \vdots \\ y_{16} \end{pmatrix} &= L \circ S \begin{pmatrix} x_1 \\ \vdots \\ x_{16} \end{pmatrix} = \begin{pmatrix} M_1 & M_2 & \cdots & M_{16} \end{pmatrix} \begin{pmatrix} S_1(x_1 \oplus rk_1) \\ \vdots \\ S_{16}(x_{16} \oplus rk_{16}) \end{pmatrix} \\ &= S_1(x_1 \oplus rk_1) M_1 \oplus S_2(x_2 \oplus rk_2) M_2 \oplus \cdots \oplus S_{16}(x_{16} \oplus rk_{16}) M_{16} \end{aligned}$$

* Corresponding author (email: salt@kookmin.ac.kr)

$$= S_1(x_1 \oplus rk_1) \begin{pmatrix} M_{1,1} \\ \vdots \\ M_{1,16} \end{pmatrix} \oplus \cdots \oplus S_{16}(x_{16} \oplus rk_{16}) \begin{pmatrix} M_{16,1} \\ \vdots \\ M_{16,16} \end{pmatrix}$$
$$= SL_1(x_1) \oplus \cdots \oplus SL_{16}(x_{16}).$$

Note that $SL_j(x_j)$ is an 8-bit to 128-bit table for $1 \leqslant j \leqslant 16$.

In a whitebox implementation, it is common to apply external input and output encodings that protect plaintexts and ciphertexts from being exposed. In Xu et al.'s design [7], external encodings are chosen as follows:

• The input encoding F is a linear transformation after a byte-wise random bijection. $F : (x_1, \ldots, x_{16}) \mapsto (y_1, \ldots, y_{16})$ by

$$\begin{pmatrix} y_1 \\ \vdots \\ y_{16} \end{pmatrix} = \begin{pmatrix} LF_{1,1} & LF_{1,2} & \cdots & LF_{1,16} \\ \vdots & \ddots & & \vdots \\ LF_{16,1} & LF_{16,2} & \cdots & LF_{16,16} \end{pmatrix} \begin{pmatrix} F_1(x_1) \\ \vdots \\ F_{16}(x_{16}) \end{pmatrix}.$$

• The output encoding G also consists of a linear transformation and a byte-wise bijection but in the reverse order of F. $G : (y_1, \ldots, y_{16}) \mapsto (z_1, \ldots, z_{16})$ by

$$\begin{pmatrix} z_1 \\ \vdots \\ z_{16} \end{pmatrix} = \begin{pmatrix} G_1\left(LG_{1,1}y_1 \oplus \cdots \oplus LG_{1,16}y_{16}\right) \\ \vdots \\ G_{16}\left(LG_{16,1}y_1 \oplus \cdots \oplus LG_{16,16}y_{16}\right) \end{pmatrix}.$$

The whitebox implementation of an SLT cipher is denoted by RBO-WBSLT which maps encoded plaintext $enPT$ to encoded ciphertext $enCT$. The encryption process using RBO-WBSLT is depicted as

$$PT \xrightarrow{F^{-1}} enPT \rightarrow \boxed{\text{RBO-WBSLT}} \rightarrow enCT \xrightarrow{G^{-1}} CT.$$

The transformations F^{-1} and G^{-1} are executed under an environment protected by the obfuscation technique. RBO-WBSLT uses two types of internal encodings.

• The byte-wise random permutation $P = P_1 \| \cdots \| P_{16}$ is defined as

$$P(x_1, \ldots, x_{16}) = P_1(x_1) \| \cdots \| P_{16}(x_{16}).$$

The inverse permutation P^{-1} of P is denoted by Q.

• An invertible linear transformation called mixing bijection $MB : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ over $GF(2)$ and its inverse MB^{-1} are used as a part of 8-bit to 128-bit tables.

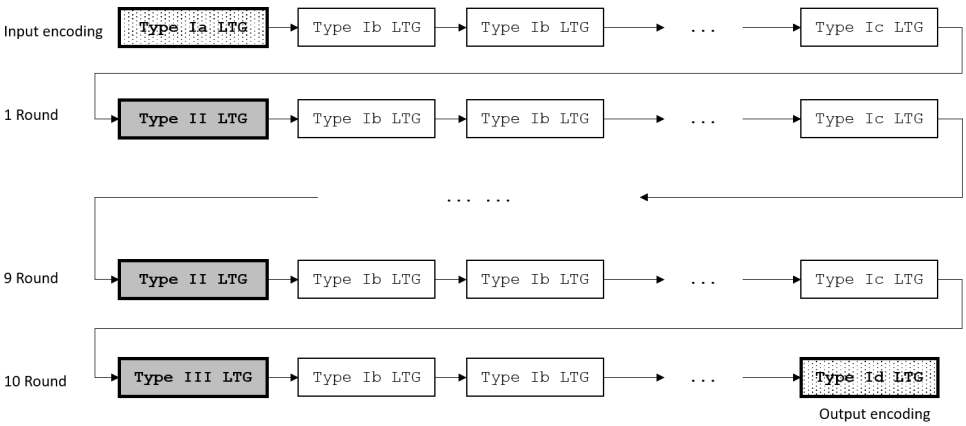

Figure A1 Structure of RBO-WBSLT.

By adding byte-wise nonlinear encodings before and after each round, we can express one round for a fixed round key $rk = (rk_1, \ldots, rk_{16})$, which maps $X = (x_1, \ldots, x_{16}) \mapsto Y = (y_1, \ldots, y_{16})$ as

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_{16} \end{pmatrix} = Q \circ \bigoplus_{i=1}^{16} SL_i(P_i(x_i) \oplus rk_i),$$

where P_i $(1 \leqslant i \leqslant 16)$ is a nonlinear 8-bit bijection and $Q = Q_1 \| \cdots \| Q_{16}$ is a parallel nonlinear bijection layer. To express the encryption process as function compositions, we require the cancel-out relations $Q^{-1}_{prev\ round} = P_{next\ round}$ for adjacent

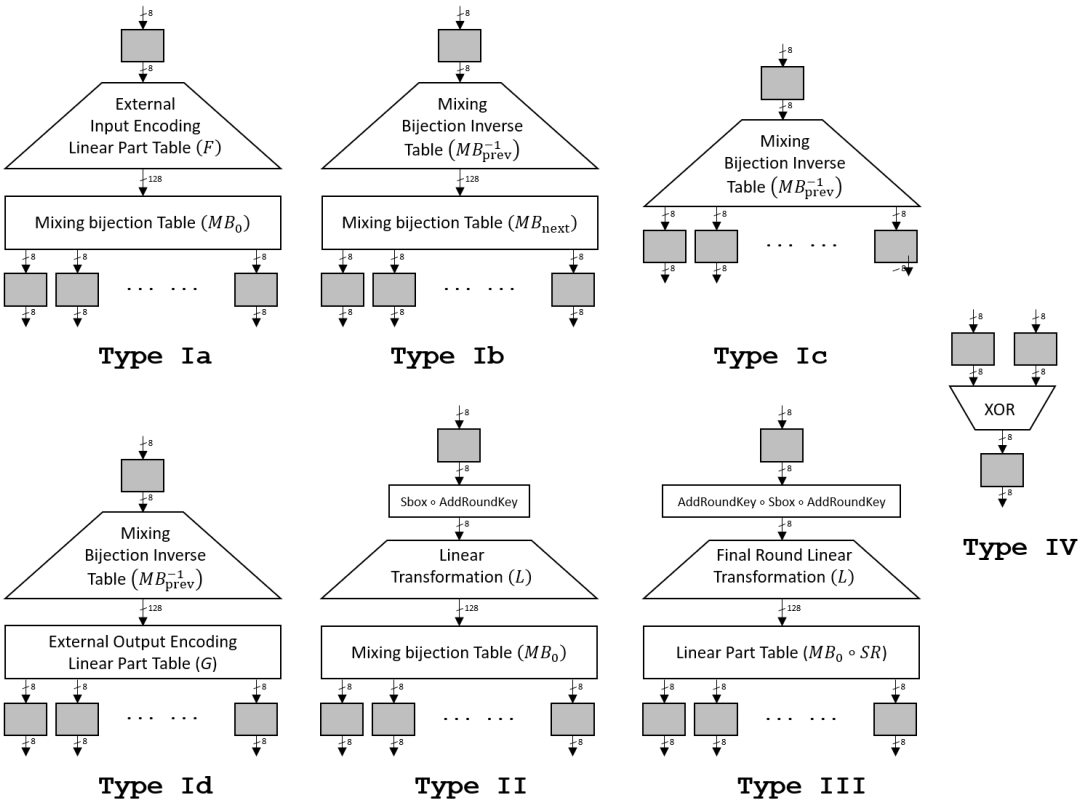

Figure A2 Tables for RBO-WBSLT.

rounds. By inserting another random bijection R in the middle, we can construct a lookup table group (LTG) for a round.

$$Y = \underbrace{Q \circ \bigoplus_{i=1}^{16} R^{-1}}_{\text{XOR layer}} \circ \underbrace{R \circ SL_i(P_i(x_i) \oplus rk_i)}_{\text{8 bit to 128 bit table}} = XOR \circ WT(X),$$

where XOR represents a layer of encoded exclusive-or tables and WT consists of 16 transforms that map a byte to a 128-bit intermediate state in parallel. To implement a round using LTG, we need sixteen 8-bit to 128-bit tables for WT and 15×16 XOR tables that map 16 bits to 8 bits, respectively. An XOR table is depicted as a "Type IV" table in Figure A2, where grey boxes represent byte-wise random bijections for the whitebox implementation.

Additionally, we can insert encoded linear transformations between XOR and WT:

$$Y = XOR \circ WT(X) = \underbrace{XOR \circ MB^{-1} \circ R^{-1}}_{\text{part 1}} \circ \underbrace{R \circ MB \circ WT(X)}_{\text{part 2}},$$

where R and MB can be interpreted as a substitution layer and linear layer, respectively. Note that "part 1" and "part 2" also have the encoded SLT structure, which can be implemented by an LTG. By inserting more MBs into the round function, Xu et al. argued that it is difficult to distinguish the key-dependent part from other parts. In Figure A1, each row represents a round, all boxes are LTG implementations of SLT, and only dark-grey boxes are key-dependent tables among them. Each box uses Type Ia, Type Ib, Type Ic, Type Id, Type II, or Type III in Figure A2. Particularly, we denote a key-dependent LTG by K-LTG. In other words, LTG layers of Type I and Type III are called K-LTGs. In Figure A1, the number of boxes in a row (in a round) is randomly chosen such that it is infeasible to determine the start or end positions of a round. However, this is not true, as shown in Appendix B.

Appendix A.3 Design rationale and security claim

Except for Table IV, all the tables in Figure A2 have the same structure; they map a byte to a 128-bit state. Given a table from among Type I to Type III, one cannot identify the type of the table. That is, it is not possible to determine the type of tables by using only their inputs and outputs. For example, a Type II table generated by the round key rk can be interpreted as a Type Ic table with a certain random encoding at the beginning.

By inserting a sufficient number of tables of Type Ib and Type Ic in the middle of the encryption process at random, Xu et al. expect that known attacks against whitebox cryptography cannot be applied to RBO-WBSLT. As a concrete example,

RBO-WBAES(10,500) with 500 LTG layers for 10 round AES is suggested. In that case, the number of possible locations for 4 consecutive *key-dependent* LTGs is approximately

$$\binom{500}{4} \approx 2^{31}.$$

If we consider the number of locations for the first 4 LTGs, then the complexity can be slightly smaller at $\binom{500-7}{4}$ but still approximately 2^{31}. With the correct locations for 4 consecutive LTGs, we can mount Lepoint et al.'s attack [6], which determines the encryption key with a complexity of 2^{22}.

In summary, RBO-WBAES(10,500) is designed to withstand known attacks by whitebox adversaries with a work factor of less than 2^{53}. To increase the security level of RBO-WBAES(10,500) up to 2^{90}, *irreversible key scheduling* is considered, where round keys are generated from an encryption key by using a one-way function such as the hash function or pseudo-random bit-generating algorithm.

Appendix B Cryptanalysis of Xu et al.'s Whitebox Implementations

Appendix B.1 Multiset properties

A *multiset* is an extended concept of a set that allows the multiplicity of its elements. For example, let

$$M_1 = \{a, b\},\ M_2 = \{a, a, b\},\ M_3 = \{a, b, b\},\ \text{and } M_4 = \{a, b, a\}.$$

Then, they denote the same set. However, as multisets, $M_2 = M_4$ and $M_2 \neq M_3$. We denote an intermediate state of the encryption process by a *state vector* $(x_1, x_2, \ldots, x_{16})$, where $x_i \in GF(2^8)$ $(1 \leqslant i \leqslant 16)$.

Here, it is sufficient to consider only multisets of 256 state vectors. Each byte position of state vectors in a multiset M is said to have properties P, C, E, D, and B, which are defined as follows:

- P (Permutation): all 2^8 possible values appear exactly once.
- C (Constant): it contains a single value.
- E (Even): each value occurs an even number of times (allowing no occurrences).
- D (Dual): the byte position has either the property P or property E.
- B (Balanced): the exclusive-or operation on all the values returns zero. $\bigoplus_{x \in M} x = 0$.

For instance, consider a multiset M defined as

$$\begin{aligned} M &= \{(x_1, \ldots, x_{16}) \,|\, x_2 = \cdots = x_{16} = 0\} \\ &= \{(0, 0, \ldots, 0), (1, 0, \ldots, 0), \ldots, (255, 0, \ldots, 0)\}. \end{aligned}$$

Then, the first byte has the property P, and the remaining bytes have the property C. In short, M is said to have the properties $(P, C, \ldots, C)$.

We will use the well-known observations in [4] on multiset properties in Lemma 1.

Lemma 1 (Multiset Properties)**.** Let M be a multiset of 256 state vectors. Then,

(a) Properties P, C, and E are preserved by any byte-wise bijection.
(b) Property B is preserved by any linear transformation.
(c) Properties $(D, \ldots, D)$ are preserved by any layer of byte-wise bijections.
(d) Properties $(D, \ldots, D)$ are transformed into properties $(B, \ldots, B)$ by any linear transformation.
(e) Properties $(P, C, \ldots, C)$ are transformed into properties $(D, \ldots, D)$ by any linear transformation.

We define SAS and SASAS structures as shown in Figure B1. The "S layer" is defined as a parallel nonlinear bijection layer such as $S = S_1 \| \cdots \| S_n$, and the "A layer" represents a linear or affine transformation. By using Lemma 1, we can easily derive the following theorem, which shows how multiset properties are transformed by SAS and SASAS structures:

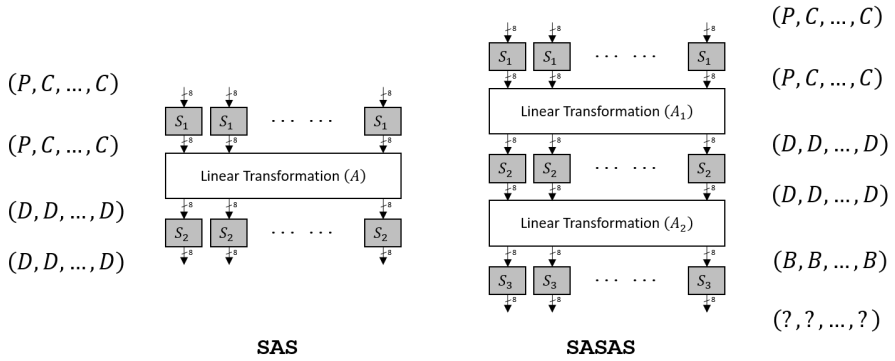

Figure B1 Multiset propagation property of SAS and SASAS structures.

Theorem 1 (Distinguishing SAS and SASAS)**.** Let a multiset M of 256 state vectors have the properties $(P, C, \ldots, C)$. Then, if we use M as input to SAS and SASAS structures, respectively,

(a) The output of the SAS structure has the properties $(D, D, \ldots, D)$ with the probability 1.

(b) The output of the SASAS structure cannot have the property P, C, D, or B with a meaningful probability.

Proof.

(a) Consider any multiset M having the properties $(P, C, \ldots, C)$ as

$$M = \{(x, c_2, \ldots, c_{16}) \,|\, 0 \leqslant x \leqslant 255, \text{ where } c_k \text{ are constants satisfying } 0 \leqslant c_k \leqslant 255 \text{ for } 2 \leqslant k \leqslant 16\}.$$

By Lemma 1 (a), the output of the first S layer has the same properties $(P, C, \ldots, C)$. Since the number of elements in M is even (256 elements), the property C can be regarded as the property D. Thus, the properties $(P, C, \ldots, C)$ can be considered as $(D, D, \ldots, D)$, which is preserved by the linear transformation. Lastly, by Lemma 1 (c), the properties $(D, \ldots, D)$ are preserved by the second S layer, which consists of byte-wise bijections.

(b) As previously proved in (b), for any input multiset $(P, C, \ldots, C)$, the output of the second S layer has the properties $(D, \ldots, D)$. Lemma 1 (d) shows that the input of the third S layer has the balanced properties $(B, \ldots, B)$. Note that, for a byte position having the balanced property B, the exclusive-or operation on all values returns zero. Unfortunately, it cannot be guaranteed that the output of the third round has a property such as P, C, D, or B. Suppose any input multiset to a byte-wise bijection has the property B, i.e., $\bigoplus_{x \in M} x = 0$. Then, the corresponding output has the property B with the approximate probability 2^{-8}, and the probabilities for P, C, and D are much less that for B. Therefore, the output of SASAS cannot have such properties with a meaningful probability, where *meaningful probability* means that the probability is greater than the probability that the event occurs by chance. □

The proof is briefly depicted in Figure B1. With the help of Theorem 1, we can easily distinguish two structures by using the 256 chosen state vectors as the input. Instead of checking the property D to confirm the SAS structure, it suffices to check the property B. Note that the output of SAS has the property B with the probability 1. However, that of SASAS has the same property with an extremely small probability of approximately $\left(2^{-8}\right)^{16}$.

Appendix B.2 Cryptanalysis of Xu et al.'s RBO-WBAES

We can observe that each table in Figure A2 has the SAS structure, and we cannot distinguish tables between Type I and Type II by using only inputs and outputs. By appending LTG layers to an LTG layer within Type I, we still observe the SAS structure because byte-wise nonlinear bijections and the MB_{prev} and MB_{next} of adjacent rounds are canceled out. However, by appending an LTG layer of Type II, we obtain an LTG layer of the SASAS structure, as shown in Figure B2. That is, we find the location of the first key-dependent LTG with Type II tables. By iterating this process as in Algorithm 1, we can eventually determine all the positions K-LTG layers of Type II and Type III.

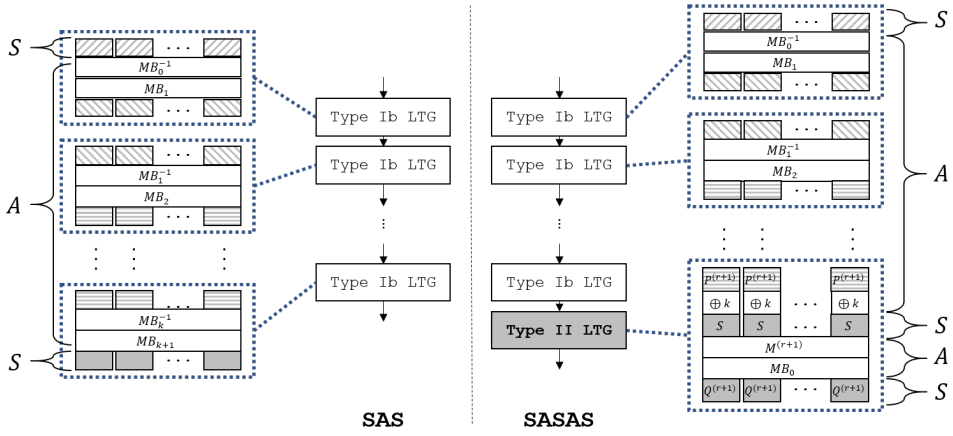

Figure B2 SAS and SASAS structures in RBO-WBSLT.

We only use a multiset of 256 state vectors to test whether the currently appended LTG is Type II or not. In fact, less than $256 \times 500 \approx 2^{17}$ encryptions are sufficient to find all the locations of key-dependent LTG layers from RBO-WBAES(10,500).

As mentioned in [7], with an LTG of Type II tables, we can extract the corresponding round key by mounting the BGE attack in [3] or more efficient attacks such as those in [2, 6].

Since the encryption key of AES is completely determined by the 16-byte round key of any round, a key-recovery attack against RBO-WBAES(10,500) succeeds with a work factor of the order of 2^{22}.

Algorithm B1 Round boundary detection algorithm

Input: Whitebox implementation of Rbo-Wbaes(10,500)
Rbo-Wbaes(10,500) $= LTG_{500} \circ \cdots LTG_2 \circ LTG_1$ and 10 K-LTGs are contained in $\{LTG_1, \ldots, LTG_{500}\}$.
Output: Starting positions for K-LTGs: $\{p_1, \ldots, p_{10}\}$

Prepare a multiset of the form $(P, C, \ldots, C)$ with 2^8 inputs to Rbo-Wbaes(10,500)
$k \leftarrow 0$, $r \leftarrow 1$;
$f \leftarrow LTG_1$;
while $r < 500$ **do**
  if (output of f) $\neq (D, \ldots, D)$ **then**
    // f has the SASAS structure (K-LTG is appended).
    $p_k \leftarrow r$;
    $k \leftarrow k + 1$;
    $f \leftarrow LTG_{r+1}$;
  else
    // f still has the SAS structure.
    $f \leftarrow LTG_{r+1} \circ f$;
  end if
  $r \leftarrow r + 1$;
end while

References

1 Specification for the advanced encryption standard (AES). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001
2 Baek C H, Cheon J H, Hong H. White-box AES implementation revisited. J Commun Netw, 2016, 18(3): 273–287
3 Billet O, Gilbert H, Ech-Chatbi C. Cryptanalysis of a white box AES implementation. In: International Workshop on Selected Areas in Cryptography, Springer, 2005. 227–240
4 Biryukov A, Shamir A. Structural cryptanalysis of SASAS. In: Proceedings of Eurocrypt 2001, Springer, 2001. 395–405
5 Chow S, Eisen P, Johnson H, Van Oorschot P C. White-box cryptography and an AES implementation. In: International Workshop on Selected Areas in Cryptography, Springer, 2003. 250–270
6 Lepoint T, Rivain M, De Mulder Y, Roelse P, Preneel B. Two attacks on a white-box AES implementation. In: International Workshop on Selected Areas in Cryptography, Springer, 2014. 265–285
7 Xu T, Wu C, Liu F, Zhao R. Protecting white-box cryptographic implementations with obfuscated round boundaries. Sci China Inform Sci, 2018, 61(3): 039103