Cryptanalysis of a Generic Class of White-Box Implementations

Wil Michiels, Paul Gorissen, and Henk D.L. Hollmann

Philips Research Laboratories, High Tech Campus 34, Eindhoven, The Netherlands
wil.michiels@philips.com

Abstract. A white-box implementation of a block cipher is a software implementation from which it is difficult for an attacker to extract the cryptographic key. Chow et al. published white-box implementations for AES and DES. These implementations are based on ideas that can be used to derive white-box implementations for other block ciphers as well. In particular, the ideas can be used to derive a white-box implementation for any substitution linear-transformation (SLT) cipher. Although the white-box implementations of AES and DES have been cryptanalyzed, the cryptanalyses published use typical properties of AES and DES. It is therefore an open question whether an SLT cipher exists for which the techniques of Chow et al. result in a secure white-box implementation. In this paper we largely settle this question by presenting an algorithm that is able to extract the key from such an implementation under a mild condition on the diffusion matrix. The condition is, for instance, satisfied by all MDS matrices. Our result can serve as a basis to design block ciphers and to develop white-box techniques that result in secure white-box implementations.

Keywords: white-box cryptography, AES, Serpent, cryptanalysis, substitution linear-transformation network, MDS matrix.

1 Introduction

The classical 'black-box' attack model used for symmetric block ciphers assumes that an attacker can at most mount chosen text attacks on the implementation. An attacker is assumed to have no access to the execution of the implementation. In practice, this model is often not realistic. Consider, for instance, a content provider who sends encrypted data to a PC platform. Then the owner of this PC may benefit from illegally distributing the key for decrypting the data to other users. In this case, it is more realistic to consider the severe 'white-box attack model' in which an attacker is assumed to have full access to and full control over the implementation of a cryptographic algorithm.

White-box cryptography is the discipline that aims at solving the problem of how to implement a cryptographic algorithm in software, such that the key cannot be extracted by a white-box attack. A software implementation of a cryptographic algorithm that has the objective to resist a white-box attack on its key is called a white-box implementation. Chow et al. present white-box implementations for the block ciphers AES and DES [4,5]. These white-box implementations are based on ideas that naturally extend to any substitution-linear transformation cipher, as defined below.

Definition 1 (Substitution-Linear Transformation Cipher (SLT cipher)). *A cipher is called an SLT cipher if it can be specified as follows. It consists of R rounds for an $R \geq 1$. A single round r is a bijective function $\mathcal{F}^{(r)}_{\mathrm{SLT}}(x_1, x_2, \ldots, x_s)$ on $\mathrm{GF}(2)^n$ with $x_i \in \mathrm{GF}(2)^m$ and $n = m \cdot s$. This function consists of the following operations. It starts with xoring an n-bit round key $k^{(r)} = (k^{(r)}_1, k^{(r)}_2, \ldots, k^{(r)}_s)$ to its input. That is, the value $y_i = k^{(r)}_i \oplus x_i$ is computed. Next, the round computes $z_i = S^{(r)}_i(y_i)$ for all y_i, where the (non-linear) invertible S-boxes $S^{(r)}_1, S^{(r)}_2, \ldots, S^{(r)}_s$ are part of the cipher specification and thus key-independent. These two steps realize confusion. The diffusion is realized by multiplying the outcome $z = (z_1, z_2, \ldots, z_s) \in \mathrm{GF}(2)^n$ of the S-boxes with an $n \times n$ invertible matrix $M^{(r)}$ over $\mathrm{GF}(2)$. This diffusion matrix is also part of the cipher specification.*

In our notation we will often omit the index r denoting the round when this value is clear from the context. White-box attacks have been published for extracting the 128-bit AES key and the 56-bit DES key from the white-box AES and DES implementations of Chow et al. [2,7,8,9,12]. The attacks use typical properties of AES and DES and do not apply to white-box implementations of other block ciphers. Hence, it remains an open question whether the white-box techniques proposed by Chow et al. can result in a secure white-box implementation for other SLT ciphers than AES, such as, for instance, Serpent [1]. In this paper we present an algorithm that, under a mild condition on the diffusion matrix, can extract the round keys from the white-box implementation of any SLT cipher. If the key scheduling algorithm is invertible, then having these round keys suffices to also derive the main key. Otherwise, we at least have a compact description of the main key. Although the time complexity of the algorithm depends on the choice of the S-boxes and the diffusion matrices, we were only able to find impractically large time complexities for unrealistic choices of these operators, e.g. linear S-boxes or diffusion matrices that are close to the identity matrix. To demonstrate the effectiveness of the proposed algorithm, we show in this paper that the algorithm can be applied successfully to AES and Serpent.

The remainder of this paper is organized as follows. In Section 2 we give a precise formulation of the result that is proved in this paper. Essential in this formulation is a specification of the information that is available to an attacker in a white-box implementation. In Sections 3-6 we present our cryptanalysis and in Section 7 we show how our ideas can be used to extract the round keys from a white-box AES and a white-box Serpent implementation. We end with a conclusion in Section 8.

2 Problem Formulation and Notation

In order to discuss the attack of a white-box implementation of an SLT cipher, we have to specify what kind of information we can obtain from such an implementation by a white-box attack. To answer this question, we briefly discuss how Chow et al. derive a white-box implementation of a block cipher.

First, they derive an implementation that, in each round of the block cipher, only performs a sequence of table lookups. The input to a lookup table is either the input to the round or it is obtained by concatenating the outputs of one or more other lookup tables. Such an implementation can be modeled by a network of lookup tables, where an arc from table T to T' means that (part of) the output of table T is used as (part of the) input to table T'.

In the design of the white-box implementation, they next obfuscate the lookup tables by encoding their inputs and outputs. Encoding the input and output of a table T with bijective functions f_{in} and f_{out}, respectively, corresponds to replacing table T by $f_{out} \circ T \circ f_{in}^{-1}$. Hence, we incorporate in T an input *de*coding and an output *en*coding. To see that such encodings realizes obfuscation, observe that encoding the input of a lookup table changes the order of its rows and that encoding the output changes the value of the rows.

The lookup tables are encoded in such a way that the functionality of the entire implementation does not change. Chow et al. show how this can be done with a combination of linear and a non-linear encodings. The non-linear encodings are applied as follows. The first tables in the network do not get an input decoding and the last ones do not get an output encoding. Furthermore, we choose the input encoding of a table, such that it matches the encoding that has been put on its input data by the tables that directly precede it in the network. To illustrate this, suppose that the entire output of a table T serves as the entire input of another table T'. We then encode the output of T by a randomly chosen encoding f and we decode the input of table T' accordingly, that is, as input decoding of table T' we employ f^{-1}. The result is that the output encoding of T and the input decoding of T' cancel out. This concludes the strategy employed by Chow et al. to add non-linear encodings. For the strategy to add additional linear encodings we refer to [4,5]. We note, however, that our cryptanalysis applies to the case that both the linear and non-linear encodings are applied.

Let $(x_1, x_2, \ldots, x_s)$ be the input to a round r of an SLT cipher, where x_i is the m-bit input to S-box S_i. Then, before applying the encodings, the network of lookup tables has the property that each word x_i is input to some lookup table T_i. For details, we refer to [4,5]. For the white-box implementation obtained after applying the (linear and non-linear) encodings, this has as a consequence that an attacker who has access to the inputs of all tables in the implementation, which holds in a white-box attack, has access to the encoded version $f_i^{(r)}(x_i)$ of each value x_i. Here $f_i^{(r)}$ is a secret bijective function that is used as input encoding for T_i. Furthermore, having access to $f_i^{(r)}(x_i)$ means that we can determine this value as well as set it to any given other value. This, for instance, implies that the attacker can choose the encoded input to a round r and next observe the effect of this for the input to a later round.

In our cryptanalysis we assume that for each value $f_i^{(r)}(x_i)$ an attacker knows the index i and the round r that are associated to this value. The index r can be derived by inspecting an execution of the white-box implementation. With respect to the value i we can in general at least limit the number of candidate values to a number that is feasible for performing an exhaustive search. We can do this by using the definition of the diffusion matrices and S-boxes and by generalizing parts of the cryptanalysis. For the sake of readability and as the difficulty of finding the value i is not considered to be the essence of the strength of a white-box implementation, we do not further discuss this problem in this paper and assume the value to be known.

This brings us at the following property, which specifies the information that is available to an attacker who tries to extract the round keys from a white-box implementation that is based on the techniques of Chow et al.

Property 1. In a white-box attack, an attacker has for each round r and for each m-bit input word x_i of round r access to the encoded version $f_i^{(r)}(x_i)$ of x_i. The attacker also knows the values of r and i that are associated with $f_i^{(r)}(x_i)$. The attacker does not know the value of the m-bit input word x_i nor the definition of the function $f_i^{(r)}$, which is an arbitrary m-bit bijective function. □

In order to formulate the main result of this paper, we need the following definitions. We note that in this paper all matrices are over GF(2).

Definition 2. *If N is an $n \times n$ matrix with $n = m \cdot s$, then we consider N to be partitioned into s vertical strips of size $n \times m$. We denote the jth strip by N_j That is,*

$$(N_j)_{x,y} = N_{x,(j-1)\cdot m+y} \ ,$$

where the rows and columns have indices in $\{1, 2, \ldots, n\}$.

Furthermore, we will consider each strip N_j to be partitioned further into s blocks $N_{i,j}$ of size $m \times m$. That is,

$$\left(N_{i,j}\right)_{x,y} = N_{(i-1)m+x,(j-1)m+y},$$

so that

$$N = (N_1 \cdots N_s) = \begin{pmatrix} N_{1,1} & N_{1,2} & \ldots & N_{1,s} \\ N_{2,1} & N_{2,2} & \ldots & N_{2,s} \\ \vdots & \vdots & \vdots & \vdots \\ N_{s,1} & N_{s,2} & \ldots & N_{s,s} \end{pmatrix}.$$

We will refer to the ith row

$$N(i) = (N_{i,1} \ N_{i,2} \ \ldots \ N_{i,s})$$

of blocks of N as the ith block row *of N.*

Definition 3. *Let N be an $n \times n$ matrix with $n = m \cdot s$. We say that a subset $U \subseteq \{1, 2, \ldots, s\}$ is a* spanning block set for block row *i if the collection of all the m-bit columns from the blocks $N_{i,j}$ with $j \in U$ spans* $\mathrm{GF}(2)^m$.

If two subsets $U, V \subseteq \{1, 2, \ldots, s\}$ with $U \cap V = \emptyset$ both represent spanning block sets for block row i, then we say that block row i has disjoint spanning block sets.

MDS (Maximum Distance Separable) matrices are often used as diffusion matrix in block ciphers because of their good diffusion properties [6,10,11]. In an MDS diffusion matrix N each block $N_{i,j}$ defines the multiplication with a non-zero element in $\mathrm{GF}(2^m)$. Hence, each block $N_{i,j}$ is non-singular, which implies that any pair of blocks from a block row i defines a spanning block set. This means that the main result of this paper, which is stated below, covers the class of SLT ciphers in which the diffusion is realized by MDS matrices.

Main Result. *Consider an SLT cipher for which the diffusion matrices have the property that all their block rows have disjoint spanning block sets. Then, given a white box implementation for this cipher that satisfies Property 1, we present an algorithm for extracting the round key of any round r with $1 < r < R$.*

The result above does not cover the first and last round of the cipher. This has the following reason. In order not to change the functionality of the white-box implementation, the input of the first round and the output of the last round cannot be encoded. However, by omitting these external encodings the white-box implementations of the first and last round become less secure. As as solution to this problem, Chow et al. propose to add the external encodings and to either undo these encodings elsewhere in the software or to include these encodings in the definition of the block cipher that is implemented. In both cases it will not only be the goal of an attacker to derive the round keys of the first and last round, but also to determine the external encodings. To simplify the discussion we exclude the attack of these rounds in this paper. We note, however, that these rounds can also be attacked. The attack is based on the following result. By applying our cryptanalysis, an attacker can determine the output encoding of the first round and the input encoding of the last round. This gives the attacker the plain output of the first round and the plain input to the last round. Using this, the first and last round can be attacked.

We end this section with the description of some notational conventions used throughout this paper.

- By abuse of notation, if N is a matrix, then the map $x \mapsto Nx$ corresponding to a matrix multiplication by N will also denoted by N.
- If T denotes a lookup table, then, by abuse of notation, we also write T to denote the function that it defines.
- We define $\oplus_c$ as the map $\oplus_c(x) = x \oplus c$. Using this, we can write the key addition of an SLT cipher as $\oplus_{k^{(r)}}$.

– Let $g_1, g_2, \ldots, g_s$ be maps on m-bit vectors, and let $n = ms$. The map $g = (g_1, \ldots, g_s)$ defined by

$$g(x) = (g_1(x_1), g_2(x_2), \ldots, g_s(x_s))$$

for each n-bit vector $x = (x_1, x_2, \ldots, x_s) \in \mathrm{GF}(2)^n$ with $x_i \in \mathrm{GF}(2)^m$ is called the *diagonal map with components* $g_1, \ldots, g_s$. When considering a diagonal map h, we will always assume that the components are maps h_i; conversely, given maps $h_1, \ldots, h_s$, we will denote the diagonal map with components h_i by h.

Remark 1. Note that if c is a vector and N a matrix, then the addition map $\oplus_c$ is always diagonal and the matrix map N is diagonal if and only if N is a *block diagonal* matrix. Here, N is called a *block diagonal* matrix if all off-diagonal blocks $N_{i,j}$, $i \neq j$, are zero. More general, it is easily verified that an affine map $\alpha : x \mapsto a \oplus Ax$ is a diagonal map if and only if A is a block diagonal matrix. Note also that the ith component of the diagonal map $x \mapsto Nx$ associated with a block diagonal matrix N is just the diagonal block $N_{i,i}$ of N.

As an example of the above conventions, we can now write the function $\mathcal{F}^{(r)}_{\mathrm{SLT}}$ describing the rth round of an SLT cipher in Definition 1 as

$$\mathcal{F}^{(r)}_{\mathrm{SLT}} = M^{(r)} \circ S^{(r)} \circ \oplus_{k^{(r)}}, \qquad (1)$$

where $M^{(r)}$ is the diffusion matrix, $S^{(r)}$ is the S-box diagonal map with as components the S-boxes $S^{(r)}_i$, and $\oplus_{k^{(r)}}$ the round-key addition map $x \mapsto x \oplus k^{(r)}$, a diagonal map with as components the maps $x_i \mapsto x_i \oplus k^{(r)}_i$.

3 Determination of the Encodings Up to an Affine Part

According to Property 1, an attacker has access to the encoded version $\tilde{x}_i = f^{(r)}_i(x_i)$ of each input word x_i of a round r, where $f^{(r)}_i$ is an unknown bijective function. In the first step of our cryptanalysis, we will show how to determine the encodings up to an affine part.

Consider a fixed round r of the white-box implementation with $1 \leq r < R$, and a block row i of the diffusion matrix M. Let sets $U = \{u_1, u_2, \ldots, u_l\}$ and $V = \{v_1, v_2, \ldots, v_{l'}\}$ be two disjoint spanning block sets for block row i of M. Without loss of generality we may assume that $U \cup V = \{1, 2, \ldots, s\}$, i.e., $l' = s - l$. This partitions the s input words of a round input into two parts: words that are input to an S-box S_i with $i \in U$ and words that are input to an S-box S_i with $i \in V$. We write $\tilde{x}'$ as the vector containing all l input words $\tilde{x}_i$ with $i \in U$ and we write $\tilde{x}''$ as the vector containing all l' input words $\tilde{x}_i$ with $i \in V$. Then the ith output word $\tilde{z}_i$ of this round r is given by $\tilde{z}_i = h(\tilde{x}', \tilde{x}'')$, where

$$h(\tilde{x}', \tilde{x}'') = f^{(r+1)}_i(\psi_U(\tilde{x}') \oplus \psi_V(\tilde{x}'')).$$

Here $\psi_U(\tilde{x}') = \bigoplus_{j \in U} \psi_j(\tilde{x}_j)$ and $\psi_V(\tilde{x}'') = \bigoplus_{j \in V} \psi_j(\tilde{x}_j)$, with

$$\psi_j(\tilde{x}_j) = M_{i,j} \circ S_j \circ \oplus_{k_j} \circ (f_j^{(r)})^{-1}(\tilde{x}_j).$$

Note that by Property 1 an attacker has access to function h, but not, for instance, to functions ψ_j. In what follows, we denote the range of a function g by $\mathrm{im}(g)$. Now $\mathrm{im}(\psi_j)$ is the vector space spanned by the columns of matrix $M_{i,j}$. Hence, as U defines a spanning block, we have $\mathrm{im}(\psi_U) = \mathrm{GF}(2)^m$. Similarly, we have $\mathrm{im}(\psi_V) = \mathrm{GF}(2)^m$. In other words, both ψ_U and ψ_V are surjective on the vector space $\mathrm{GF}(2)^m$. In the full paper we prove Theorem 1 below, which bounds the time complexity for the construction of sets W_U and W_V which are mapped bijectively onto $\mathrm{GF}(2)^m$ by ψ_U and ψ_V, respectively.

Theorem 1. *In $\mathcal{O}(s + m \cdot 2^m)$ time, we can construct sets W_U and W_V, with $|W_U| = |W_V| = 2^m$, such that*
(i) for each fixed $\tilde{x}''$, the map $\tilde{x}' \mapsto h(\tilde{x}', \tilde{x}'')$ is a bijection on W_U, and
(ii) for each fixed $\tilde{x}'$, the map $\tilde{x}'' \mapsto h(\tilde{x}', \tilde{x}'')$ is a bijection on W_V.

Let h_c denote the bijective function $\tilde{x}' \mapsto h(\tilde{x}', c)$ from W_U onto $\mathrm{GF}(2)^m$. Let $c_1, c_2 \in W_V$, and put $d = \psi_V(c_1) \oplus \psi_V(c_2)$. Now if $\tilde{z}_i = h_{c_2}(\tilde{x}') = f_i^{(r+1)}(\psi_U(\tilde{x}') \oplus \psi_V(c_2))$, then $\psi_U(\tilde{x}') = \psi_V(c_2) \oplus (f_i^{(r+1)})^{-1}(\tilde{z}_i)$, and hence

$$h_{c_1} \circ h_{c_2}^{-1}(\tilde{z}_i) = f_i^{(r+1)}(\psi_U(\tilde{x}') \oplus \psi_V(c_1)) = f_i^{(r+1)} \circ \oplus_d \circ (f_i^{(r+1)})^{-1}(\tilde{z}_i).$$

Now fix $c_1 \in W_V$. Then to each $c_2 \in W_V$ there corresponds a unique $d \in \mathrm{GF}(2)^m$. Hence by letting $\tilde{x}'$ run through W_U, we can construct for each d in $\mathrm{GF}(2)^m$ a lookup table for the function $f_i^{(r+1)} \circ \oplus_d \circ (f_i^{(r+1)})^{-1}$. We can then use the following result of Billet et al. [2] to determine $f_i^{(r+1)}$ up to an affine part.

Theorem 2. *Let $f_i^{(r+1)}$ be an arbitrary bijective function on $\mathrm{GF}(2^m)$. Suppose that the set of functions $\{f_i^{(r+1)} \circ \oplus_d \circ (f_i^{(r+1)})^{-1} \mid d \in \mathrm{GF}(2^m)\}$ is given by means of lookup tables. Then we can construct in $\mathcal{O}(2^{3m})$ time a function $g_i^{(r+1)}$ for which the map $\alpha_i^{(r+1)} = g_i^{(r+1)} \circ f_i^{(r+1)}$ is affine.*

Combining Property 1 with the above theorem shows that for any input word x_i, an attacker can obtain the value of the affine map $\alpha_i^{(r+1)}(x_i) = g_i^{(r+1)} \circ f_i^{r+1}(x_i)$. So we have the following.

Property 2. In a white-box attack, an attacker has for each round r with $2 \leq r \leq R$ and for each m-bit input word x_i of round r access to the encoded version $\tilde{x}_i = \alpha_i^{(r)}(x_i)$ of x_i. Here, $\alpha_i^{(r)}(x_i) = A_i^{(r)} x \oplus a_i^{(r)}$ is an m-bit affine function. The attacker also knows the values of r and i that are associated with $f_i^{(r)}(x_i)$. $\square$

4 Transformation into Table Network

From Property 2 it follows that upon completion of the first step of the cryptanalysis, an attacker has for each round r with $1 < r < R$ access to the input-output behavior of the function

$$\mathcal{G}^{(r)}_{\mathrm{SLT}} = \alpha^{(r+1)} \circ \mathcal{F}^{(r)}_{\mathrm{SLT}} \circ (\alpha^{(r)})^{-1}. \tag{2}$$

As before, fix a round r. Then we have $\mathcal{G}_{\mathrm{SLT}} = \alpha^{(r+1)} \circ \mathcal{F}_{\mathrm{SLT}} \circ (\alpha^{(r)})^{-1}$, and by (1), we have that $\mathcal{F}_{\mathrm{SLT}} = M \circ S \circ \oplus_k$. Hence, if we write $N = A^{(r+1)}M$ and $R = S \circ \oplus_k \circ (\alpha^{(r)})^{-1}$, then we have that

$$\mathcal{G}_{\mathrm{SLT}}(\tilde{x}) = \oplus_{a^{(r+1)}} \circ N \circ R(\tilde{x}) = a^{(r+1)} \oplus \bigoplus_{j=1}^{s} N_j \circ R_j(\tilde{x}_j), \tag{3}$$

where N_j denotes the jth strip of matrix N and R_j denotes the jth component of diagonal map R.

We will first derive an implementation of $\mathcal{G}_{\mathrm{SLT}}$ involving s lookup tables only. More specifically, we will define tables $T_1, T_2, \ldots, T_s$ such that $\mathcal{G}_{\mathrm{SLT}}(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_s)$ equals $\bigoplus_{i=1}^{s} T_i(\tilde{x}_i)$. To this end, we define T_j as

$$T_j(\tilde{x}_j) = \begin{cases} \mathcal{G}_{\mathrm{SLT}}((\tilde{x}_1, 0, \ldots, 0)), & \text{if } j = 1; \\ \mathcal{G}_{\mathrm{SLT}}((0, \ldots, 0, \tilde{x}_j, 0, \ldots, 0)) \oplus \mathcal{G}_{\mathrm{SLT}}(0), & \text{otherwise.} \end{cases}$$

We have $\mathcal{G}_{\mathrm{SLT}}(0) = \oplus_{a^{(r+1)}} \circ N \circ R(0)$ and, for all $j \geq 1$,

$$\mathcal{G}_{\mathrm{SLT}}((0, \ldots, 0, \tilde{x}_j, 0, \ldots, 0)) = \mathcal{G}_{\mathrm{SLT}}(0) \oplus N_j \circ R_j(0) \oplus N_j \circ R_j(\tilde{x}_j).$$

Hence,

$$T_j(\tilde{x}_j) = \begin{cases} a^{(r+1)} \oplus N_1 \circ R_1(\tilde{x}_1) \oplus \bigoplus_{i=2}^{s} N_i \circ R_i(0), & \text{if } j = 1; \\ N_j \circ R_j(0) \oplus N_j \circ R_j(\tilde{x}_j), & \text{for } j = 2, \ldots, s, \end{cases} \tag{4}$$

and hence we immediately see that $\mathcal{G}_{\mathrm{SLT}}(\tilde{x}) = \bigoplus_{i=j}^{s} T_j(\tilde{x}_j)$ as desired.

5 Transformation into SAT Cipher

In an SLT cipher, we can merge a key-addition operation into the S-box operation that succeeds it. The resulting S-box is then given by $S_i \circ \oplus_{k_i}$. Hence, an SLT cipher can be viewed as a generic SAT cipher, which is defined as follows.

Definition 4 (Generic Substitution-Affine Transformation Cipher (generic SAT cipher)). *A cipher is called a generic SAT cipher if it can be specified as follows. It consists of R rounds for an $R \geq 1$. A single round r is a bijective function $\mathcal{F}_{\mathrm{gen-SAT}}(x_1, x_2, \ldots, x_s)$ on $\mathrm{GF}(2)^n$ with $x_i \in \mathrm{GF}(2)^m$ and $n = m \cdot s$.*

A round consists of the following operations. First, the values $y_i = Q_i^{(r)}(x_i)$ are computed for all input words x_i, where the specification of an S-box $Q_i^{(r)}$ is derived from the key. Next, an invertible affine function $\epsilon^{(r)}(y) = E^{(r)} \cdot y \oplus e^{(r)}$ is applied to the outcome $y = (y_1, y_2, \ldots, y_s) \in \mathrm{GF}(2)^n$ of the S-boxes. The specification of this affine function $\epsilon^{(r)}$ is also derived from the key.

So the round function $\mathcal{F}_{\text{gen}-\text{SAT}}$ of a generic SAT cipher can be written as $\mathcal{F}_{\text{gen}-\text{SAT}} = \epsilon \circ Q$, where ϵ is affine and Q is a diagonal map, with both Q and ϵ fully specified by the key.

We can consider $\mathcal{G}_{\text{SLT}}$ as a generic SAT cipher. Indeed, according to (3), $\mathcal{G}_{\text{SLT}} = \theta \circ R$, where $\theta = \oplus_{a^{(r+1)}} \circ N$ is affine and R is a diagonal map. However, since the functions θ and R are not accessible to an attacker, this form is not suitable for cryptanalysis. In what follows, we will develop an alternative specification for $\mathcal{G}_{\text{SLT}}$ as a SAT cipher $\mathcal{G}_{\text{SLT}} = \epsilon \circ Q$, with an affine function ϵ and a diagonal map Q that are both accessible to an attacker. We will then use this expression to attack the round key of the original STL cipher $\mathcal{F}_{\text{SLT}}$.

We begin with a simple observation. Since both the diffusion matrix M and the diagonal matrix $A^{(r+1)}$ from the affine map $\alpha^{(r+1)}$ are invertible, the matrix $N = A^{(r+1)}M = (N_1 \cdots N_s)$ is also invertible, and hence the columns of each of the $n \times m$ matrices N_j are also independent. Let U_j denote the m-dimensional vector space spanned by the columns of N_j. Next, we consider again Expression (4) for the lookup tables T_j. Putting $w_1 = a^{(r+1)} \oplus N \circ R(0) \oplus N_1 \circ R_1(0)$ and $w_j = N_j \circ R_j(0)$ for $j = 2, \ldots, s$, it follows from (4) that $\mathrm{im}(T_j) = w_j \oplus U_j$. Hence the 2^m rows of lookup table T_j together comprises all vectors from $w_j \oplus U_j$.

Now, select an arbitrary row v_j from each table T_j, and define

$$e = \bigoplus_{j=1}^{s} v_j.$$

Note that for each $\tilde{x}_j$, we have that $v_j \oplus T_j(\tilde{x}_j) \in U_j$. Indeed, since both v_j and $T_j(\tilde{x}_j)$ are rows of T_j, there are u_j and u'_j in U_j such that $v_j = w_j \oplus u_j$ and $T_j(\tilde{x}_j) = w_j \oplus u'_j$. But then $v_j \oplus T_j(\tilde{x}_j) = u_j \oplus u'_j \in U_j$, as claimed. Next, by selecting words $\tilde{x}_{j,1}, \ldots, \tilde{x}_{j,m}$ for which the vectors $e_{j,i} = v_j \oplus T_j(\tilde{x}_{j,i})$ in U_j are independent, we can construct a basis $e_{j,1}, \ldots, e_{j,m}$ for each U_j. We use these bases to define the submatrices E_j of $E = (E_1, E_2, \ldots, E_s)$ as

$$E_j = (e_{j,1} \cdots e_{j,m}).$$

Since the m columns of E_j span U_j and since $T_j(\tilde{x}_j) \in v_j \oplus U_j$ for each $\tilde{x}_j \in \mathrm{GF}(2)^m$, there is a vector $Q_j(\tilde{x}_j) \in \mathrm{GF}(2)^m$ such that

$$T_j(\tilde{x}_j) = v_j \oplus E_j Q_j(\tilde{x}_j).$$

We will consider Q_j as a map from $\mathrm{GF}(2)^m$ to $\mathrm{GF}(2^m)$. As all rows of table T_j are different, this map is a bijection. Now let $Q = (Q_1, \ldots, Q_s)$ be the diagonal map with components Q_j, and define the affine map ϵ by $\epsilon = \oplus_e \circ E$. By the above analysis, we have that

$$\epsilon \circ Q(\tilde{x}) = e \oplus E \circ Q(\tilde{x}) = \bigoplus_{j=1}^{s} v_j \oplus \bigoplus_{j=1}^{s} E_j Q_j(\tilde{x}_j) = \bigoplus_{j=1}^{s} T_j(\tilde{x}_j) = \mathcal{G}_{\text{SLT}}(\tilde{x}),$$

hence $\mathcal{G}_{\text{SLT}} = \epsilon \circ Q$ is a representation of $\mathcal{G}_{\text{SLT}}$ as a SAT cipher where both the affine map ϵ and the diagonal map Q are explicitly known and accessible to an attacker.

6 Extracting the Key

In this chapter we describe the last step of our cryptanalysis. We adopt the following strategy. First, we derive a relation between the S-boxes $S_i^{(r)}$ of the white-boxed SLT cipher and the S-boxes $Q_i^{(r)}$ of the generic SAT cipher that we constructed in Section 5. This relation will be of the form $Q_i^{(r)} = \gamma_i^{(r)} \circ S_i^{(r)} \circ \delta_i^{(r)}$ for affine functions $\gamma_i^{(r)}, \delta_i^{(r)}$. The diagonal map $\delta^{(r)} = (\delta_1^{(r)}, \delta_2^{(r)}, \dots, \delta_s^{(r)})$ depends on both the round key $k^{(r)}$ of $\mathcal{F}_{\text{SLT}}^{(r)}$ and the affine encoding $\alpha^{(r)}$ that $\mathcal{G}_{\text{SLT}}^{(r)}$ puts on the input of round $\mathcal{F}_{\text{SLT}}^{(r)}$. The function $\gamma^{(r)} = (\gamma_1^{(r)}, \gamma_2^{(r)}, \dots, \gamma_s^{(r)})$ depends on the encoding $\alpha^{(r+1)}$ that $\mathcal{G}_{\text{SLT}}^{(r)}$ puts on the output of $\mathcal{F}_{\text{SLT}}^{(r)}$. By comparing the functions $\gamma^{(r-1)}$ and $\delta^{(r)}$, we can recover the key $k^{(r)}$ contained in $\delta^{(r)}$.

We now make the last step of our cryptanalysis more precise. Fix a round r. From the previous step we get S-boxes Q_j and an affine function ϵ, such that $\mathcal{G}_{\text{SLT}} = \epsilon \circ Q$. By (1) and (2), we also have that

$$\mathcal{G}_{\text{SLT}} = \alpha^{(r+1)} \circ M \circ S \circ \oplus_k \circ (\alpha^{(r)})^{-1}.$$

Since the functions ϵ, $\alpha^{(r+1)}$, M, $\oplus_k$, and $\alpha^{(r)}$ are all affine, we conclude that

$$Q = \gamma \circ S \circ \delta, \tag{5}$$

for affine functions

$$\gamma = \epsilon^{-1} \circ \alpha^{(r+1)} \circ M \tag{6}$$

and

$$\delta = \oplus_k \circ (\alpha^{(r)})^{-1}. \tag{7}$$

Note that both γ and δ are diagonal maps. For δ this is true because $\alpha^{(r)}$ is a diagonal map. For γ this property follows from (5) and the observation that δ, Q, and S are all diagonal maps. Biryukov et al. [3] present an algorithm for efficiently determining the set Γ_i of all pairs (γ_i, δ_i) satisfying $Q_i = \gamma_i \circ S_i \circ \delta_i$. So we can use this algorithm to determine the set Γ consisting of all pairs (γ, δ) that satisfy (5). Note that this set Γ contains the pair (γ, δ) that satisfies all of (5), (6), and (7). To complete our cryptanalysis it now suffices to solve the following two problems.

- Which pairs $(\gamma^{(r-1)}, \delta^{(r-1)}) \in \Gamma^{(r-1)}$ and $(\gamma^{(r)}, \delta^{(r)}) \in \Gamma^{(r)}$ of affine functions satisfy (6) and (7)?
- If we have the pairs $(\gamma^{(r-1)}, \delta^{(r-1)}) \in \Gamma^{(r-1)}$ and $(\gamma^{(r)}, \delta^{(r)}) \in \Gamma^{(r)}$ of affine functions that satisfy (6) and (7), how can we derive round key $k^{(r)}$ from this?

Observe that the former problem need not be solved completely. If we can limit the number of candidate pairs to a value ℓ, then we can apply an algorithm for the second problem to all ℓ candidate solutions to obtain ℓ candidate round keys. The correct round key can next be derived by exhaustive search.

By (6) for round $r-1$ and (7) for round r, we have that $\epsilon^{(r-1)} \circ \gamma^{(r-1)} = \alpha^{(r)} \circ M^{(r-1)}$ and $\alpha^{(r)} = (\delta^{(r)})^{-1} \circ \oplus_{k^{(r)}}$, and hence $\delta^{(r)} \circ \epsilon^{(r-1)} \circ \gamma^{(r-1)} = \oplus_{k^{(r)}} \circ M^{(r-1)}$. So if we let $\gamma^{(t)} = \oplus_{c^{(t)}} \circ C^{(t)}$ and $\delta^{(t)} = \oplus_{d^{(t)}} \circ D^{(t)}$, then we obtain that

$$d^{(r)} \oplus D^{(r)} e^{(r-1)} \oplus D^{(r)} E^{(r-1)} C^{(r-1)} y \oplus D^{(r)} E^{(r-1)} c^{(r-1)} = k^{(r)} \oplus M^{(r-1)} y,$$

for all y. For this equality to hold, the constant parts as well as the linear parts are the same in both sides of the equation. This implies that

$$D^{(r)} E^{(r-1)} C^{(r-1)} = M^{(r-1)}$$

and that the round key $k^{(r)}$ is given by

$$k^{(r)} = d^{(r)} \oplus D^{(r)} e^{(r-1)} \oplus D^{(r)} E^{(r-1)} c^{(r-1)}.$$

The above analysis now leads to the algorithm described in Fig. 1 for finding $k^{(r)}$.

Known: Q, E, S, M.

- Step 1: For a particular pair $(r-1, r)$ of successive rounds, construct for each S-box S_i the set
$$\Gamma_i = \{(\gamma_i, \delta_i) \mid Q_i = \gamma_i \circ S_i \circ \delta_i \wedge \gamma_i, \delta_i \text{ affine}\}$$
and let Γ be such that $(\gamma, \delta) \in \Gamma$ if $(\gamma_i, \delta_i) \in \Gamma_i$ for all i.
- Step 2: Construct the subset $\Lambda^{(r)} \subseteq \Gamma^{(r-1)} \times \Gamma^{(r)}$ of pairs of affine functions $(\gamma^{(r-1)}, \delta^{(r-1)}) \in \Gamma^{(r-1)}$ and $(\gamma^{(r)}, \delta^{(r)}) \in \Gamma^{(r)}$ such that
$$D^{(r)} E^{(r-1)} C^{(r-1)} = M^{(r-1)}, \tag{8}$$
where matrices $C^{(r-1)}$ and $D^{(r)}$ define the linear part of $\gamma^{(r-1)}$ and $\delta^{(r)}$, respectively.
- Step 3: The round key $k^{(r)}$ of round r is contained in the set
$$K^{(r)} = \left\{ D^{(r)} E^{(r-1)} c^{(r-1)} \oplus D^{(r)} e^{(r-1)} \oplus d^{(r)} \mid (\gamma^{(r-1)}, \delta^{(r-1)}, \gamma^{(r)}, \delta^{(r)}) \in \Lambda^{(r)} \right\}.$$

Fig. 1. Basic algorithm for finding the round key of a round r

For implementing Step 1 of the algorithm, we already referred to [3]. We now describe how Step 2 can be implemented.

6.1 Solving the Linear Equivalence Problem for Matrices

Step 2 of the algorithm of Fig. 1 deals with the matrices C and D specifying the linear parts of the affine m-bit diagonal maps $\gamma = \oplus_c \circ C$ and $\delta = \oplus_d \circ D$. Note that as observed in Remark 1, C and D are block diagonal matrices.

Definition 5. *Let $X = X_1 \times X_2 \times \ldots \times X_s$, where X_i consists of $m \times m$ matrices. Then we denote by $\mathcal{D}(X)$ the collection of all block diagonal matrices with ith diagonal block contained in X_i, for all i.*

We can now formulate the problem of Step 2 as an instance of the Linear Equivalence Problem of Matrices (LEPM) defined below.

Definition 6 (Linear Equivalence Problem of Matrices (LEPM)). *A problem instance is defined by (M, E, X, Y) for invertible $n \times n$ matrices M and E and sets $X = X_1 \times X_2 \times \ldots \times X_s$ and $Y = Y_1 \times Y_2 \times \ldots \times Y_s$, where X_i and Y_j contain invertible $m \times m$ matrices and $n = m \cdot s$. Find all pairs of block-diagonal $n \times n$ matrices $(C, D) \in \mathcal{D}(X) \times \mathcal{D}(Y)$ such that $M = D \cdot E \cdot C$.*

In Fig. 2 we describe an algorithm for solving LEPM. The algorithm gradually reduces the sets X_i and Y_j, as follows. If a pair $(C, D) \in \mathcal{D}(X) \times \mathcal{D}(Y)$ satisfies $M = D \cdot E \cdot C$, then $M_{i,j} = D_i E_{i,j} C_j$ holds for all i, j. So if for some $C_j \in X_j$ there does not exist a $D_i \in Y_i$ for which $M_{i,j} = D_i E_{i,j} C_j$, then C_j can never be used as jth component in C, and so can be removed from X_j. A similar argument can be used to remove a matrix D_i from a set Y_i.

We proceed with such removal steps until no more removals are possible. If some set X_j or some set Y_i is empty, then the LEPM problem has no solution. Next, if all sets X_j and Y_i contain exactly one linear mapping, then the only candidate solution to the LEPM problem instance is the solution defined by these linear mappings; moreover, since no X_j or Y_i was further reduced, this solution must indeed be valid. So in this case, the LEPM problem is solved.

On the other hand, suppose that a set X_j or a set Y_i exists that contains more than one linear mapping. If all X_j have size one, then C is uniquely determined, and hence $D = E^{-1}C^{-1}M$ is also uniquely determined. As a consequence, all sets Y_i must also have size one. So we may assume without loss of generality that some set X_j has size bigger than one. In that case, for each matrix C_j in X_j, we rerun the algorithm with X_j replaced by the set $X'_j = \{C_j\}$. Obviously, in this way all solutions are found.

To know whether the algorithm presented is effective for attacking a white-box implementation, we have to know an upper bound on the number of solutions returned and the number of recursive invocations. The former number is related to the cardinality of the set K of candidate round keys in the algorithm of Fig. 1. The latter number determines the time complexity of the algorithm of Fig. 2. The problem is that we do not want to answer the question for one particular white-box implementation of a block cipher, but for any white-box implementation of that block cipher. Hence, we want to derive upper bounds on these numbers that only depend on the block cipher specification and not, for instance, on the encodings put on the input and output of a round $\mathcal{F}_{\text{SLT}}$ by the white-box implementation. The following theorem, which is proved in the full version of this paper, can be used to derive such bounds.

Theorem 3. *For a round r of an SLT cipher, let $I = (M, E, X, Y)$ be the problem instance of LEPM that is associated with the cryptanalysis of its white-box*

```
algorithm LEPM_solver($X, Y$)

begin
    repeat
        for all $X_j$ do
            for all $C_j \in X_j$ do
                if $\neg\exists_{D_i \in Y_i} M_{i,j} = D_i \cdot E_{i,j} \cdot C_j$ then
                    $X_j := X_j \setminus \{C_j\}$;
        for all $Y_i$ do
            for all $D_i \in Y_i$ do
                if $\neg\exists_{C_j \in X_j} M_{i,j} = D_i \cdot E_{i,j} \cdot C_j$ then
                    $Y_i := Y_i \setminus \{D_i\}$;
    until $X$ and $Y$ do not change;
    if a set $X_j$ or $Y_i$ is empty then
        return $\emptyset$;
    else if $\forall_j |X_j| = 1 \wedge \forall_i |Y_i| = 1$ then
        return $\{(C, D)\}$ with $C_j \in X_j$ and $D_i \in Y_i$;
    else /* case $\exists_j |X_j| > 1$ */
        select smallest $j$ with $|X_j| > 1$;
        return $\bigcup_{C_j \in X_j}$ LEPM_solver($X(X_j = \{C_j\}), Y$);
end;
```

Fig. 2. Algorithm for solving LEPM problem in pseudo code. In the algorithm $X(X_j = \{C_j\})$ denotes X, where X_j is replaced by $\{C_j\}$.

implementation. Furthermore, let $I' = (M, M, X', Y')$ be the problem instance in which X'_i and Y'_i are given by

$$X'_i = \{L \mid S_i^{(r-1)} = \lambda \circ S_i^{(r-1)} \circ \phi \text{ with } \lambda : x \mapsto l \oplus Lx \text{ and } \phi \text{ affine}\}$$

and

$$Y'_i = \{P \mid S_i^{(r)} = \lambda \circ S_i^{(r)} \circ \phi \text{ with } \lambda \text{ and } \phi : x \mapsto p \oplus Px \text{ affine}\}.$$

Then, applying the algorithm of Fig. 2 to I results in the same number of recursive invocations and the same number of solutions as when applying the algorithm to problem instance I'.

7 Proof of Concept

As proof of concept, we briefly discuss our cryptanalysis for attacking white-box AES and white-box Serpent. It can be verified that the diffusion matrices of both AES and Serpent satisfy the property that all their block rows have disjoint spanning blocks. Recall that this is a necessary property to perform the first step of the cryptanalysis. After applying the steps described in Sections 3-5, the cryptanalysis runs the algorithm of Fig. 1 to find a set K of candidate round keys for a given round r. The algorithm first derives for each S-box S_i the set Γ_i. For the AES S-box these sets can be shown to have a cardinality of 2040, while for the Serpent S-boxes the cardinality of these sets is either 4 or 1. Next, the algorithm solves an LEPM problem instance to find the set Λ. For AES and Serpent it can be shown that the pairs (γ_i, δ_i) from a set Γ_i satisfy the property that all affine functions γ_i have a unique linear part and that all affine functions δ_i have a unique linear part. Hence, the cardinality of set Λ is given by the number of solutions of this LEPM problem instance. Using Theorem 3 it can be proved that for any LEPM problem instance associated with a white-box implementation the algorithm does not go into recursion and that it returns only one solution. As a consequence, Λ consists of only one solution. It now follows from the third step of the algorithm of Fig. 1 that the set K of candidate round keys consists of only one solution as well. This is the round key we are looking for. The time complexity of the attack is dominated by the algorithm of Biryukov et al. [3] to determine the sets Γ_i.

8 Conclusion

Chow et al. published white-box implementations for AES and DES. As these white-box implementations have been broken, it is an interesting research direction to design a block cipher that results in a secure white-box implementation. This paper can serve as a basis for such research. In this paper we presented an algorithm for extracting the round keys from the white-box implementation of an SLT cipher in case that all block rows of the diffusion matrices of the cipher have disjoint spanning block sets. The condition on the diffusion matrices is, for instance, satisfied by all MDS matrices. Furthermore, we conjecture that our attack can be generalized to arbitrary diffusion matrices. From our result we can conclude that, unless we design new white-box techniques, SLT ciphers are less suited for white-box implementations. A weakness of SLT ciphers that is exploited by our attack is the linearity of the diffusion operator. A linear diffusion matrix is difficult to hide with non-linear encodings. Hence, a possible direction for deriving secure white-box implementations is to resort to alternative diffusion operators. Another weakness of SLT ciphers that we exploit is that, except for a key addition, all operations in the cipher are fixed (i.e., key-independent). It may help to make a larger part of the block cipher operations key-dependent.

References

1. Anderson, R.J., Biham, E., Knudsen, L.R.: Serpent: A proposal for the advanced encryption standard. In: Proceedings of the First AES Candidate Conference (1998)
2. 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)

3. 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)
4. 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)
5. Chow, S., Eisen, P., Johnson, H., van Oorschot, P.C.: White-box cryptography and an AES implementation. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 250–270. Springer, Heidelberg (2003)
6. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer, Heidelberg (2002)
7. Goubin, L., Masereel, J.-M., Quisquater, M.: Cryptanalysis of white box DES implementations. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 278–295. Springer, Heidelberg (2007)
8. Jacob, M., Boneh, D., Felten, E.W.: Attacking an obfuscated cipher by injecting faults. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 16–31. Springer, Heidelberg (2003)
9. Link, H.E., Neumann, W.D.: Clarifying Obfuscation: Improving the Security of White-Box DES. In: International Symposium on Information Technology: Coding and Computing, pp. 679–684 (2005)
10. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: The Twofish Encryption Algorithm: A 128-Bit Block Cipher. Wiley, Chichester (1999)
11. Vaudenay, S.: On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In: Proceedings of the 2nd International Workshop on Fast Software Encryption, pp. 286–297 (1995)
12. Wyseur, B., Michiels, W., Gorissen, P., Preneel, B.: Cryptanalysis of white-box DES implementations with arbitrary external encodings. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 264–277. Springer, Heidelberg (2007)