Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings

Brecht Wyseur[1], Wil Michiels[2], Paul Gorissen[2], and Bart Preneel[1]

[1] Katholieke Universiteit Leuven
Dept. Elect. Eng.-ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
{brecht.wyseur,bart.preneel}@esat.kuleuven.be

[2] Philips Research Laboratories
Prof. Holstlaan 4, 5656 AA, Eindhoven, The Netherlands
{wil.michiels,paul.gorissen}@philips.com

March 21st, 2007

Abstract. At SAC 2002, Chow *et al.* [4] presented a method for implementing the DES block cipher such that it becomes hard to extract the embedded secret key in a white-box attack context. In such a context, an attacker has full access to the implementation and the executing environment. In order to provide an extra level of security, an implementation shielded with external encodings was introduced by Chow *et al.* and improved by Link and Neumann [9]. Several attacks and improvements have been presented on the implementation without these encodings, but until February 2007, none were able to extract the shielded implementation. In this paper, we present an algorithm to extract the secret key from the implementation as presented by Link and Neumann. The cryptanalysis is a differential attack on the obfuscated round, with the advantage that it is independent of the shielding external encodings. Therefore it can be applied to any existing white-box DES implementation. We also implemented this cryptanalysis which runs with a time complexity of 2^{14}, and has a negligible space complexity.

1 Introduction

White-box cryptography attempts to protect secret keys embedded into the software implementation of a block cipher. The attack model for these implementations is defined as the white-box attack model. In this model, an attacker is considered to have full control over the implementation and its execution environment. This model is the opposite of the black-box attack model in which an attacker can only use the input and output behaviour of cryptographic algorithms in order to find the secret key.

For the black-box model, several strong cryptographic block ciphers have been proposed, such as DES *(Data Encryption Standard)*, its successor AES

(Advanced Encryption Standard) and many others. attacks have been presented on reduced round versions. Cipher designers aim to reduce the number of rounds, for which a cipher is secure, while cryptanalysists try to construct an attack on as many rounds as possible. For AES-128 and AES-192, a 7 and 8 round cryptanalysis has been presented respectively (out of 10 and 12 rounds). For DES, 3 theoretical attacks already exist on the full 16 rounds, but until now no practical attacks have been found. In a white-box attack model, this game of design and cryptanalysis fails completely, since an attacker has access to the round functions, and can thus perform a cryptanalysis on a chosen part of the implementation representing a reduced number of round functions.

In 2002, Chow *et al.* [4] proposed a white-box implementation of DES. The main idea is to implement the block cipher as a network of lookup tables. All the operations of the block cipher, such as the key addition, are embedded into these lookup tables, which are then randomised to obfuscate their behaviour. This process of obfuscation obstructs attacks such as the cryptographic attacks on a reduced number of rounds, timing attacks, cache attacks (e.g., [10]) or implementation attacks [8].

Together with the white-box DES implementation proposal, Chow *et al.* [3] described a white-box AES implementation. The design principles are similar and shielding encodings are applied in the same way as they are applied to white-box DES implementations. The encoded white-box AES implementation has been cryptanalysed by Billet *et al.* [2] using algebraic properties of AES to attack the implementation on the obfuscated round functions. Several publications described cryptanalysis results of 'naked' white-box DES implementations, i.e., without the shielding (external) encodings. Chow *et al.* have indicated the weakness of the naked variant in the original paper on DES white-box [4]. Jacob *et al.* [7] and Link and Neumann [9] also presented attacks on this variant. In [9], Link and Neumann present a white-box DES implementation that has no external encodings and that is resistant to these three attacks. Both this naked variant and the implementation of Chow *et al.* with external encodings have withstood cryptanalysis.

In this paper, we describe the cryptanalysis of both these white-box DES implementations. The work presented has been conducted independently from Goubin *et al.* [6], who presented independent results at the time of writing this paper. In their paper, Goubin *et al.* briefly describe a cryptanalysis of the white-box DES implementation of Link and Neumann [9], which is then extended to a cryptanalysis of the white-box DES implementation with external encodings, through the analysis of the external input encoding. They can only deal with the case of linear external encodings; our new method works independently of the external encodings, and can thus be applied to white-box DES implementation with arbitrary external encodings.

The attack presented in this paper targets the internal behaviour of the implementation; it is a differential cryptanalysis [1] on the obfuscated round functions, which are accessible in a white-box environment, and it is independent of the definition and implementation of the external encodings. Therefore, it applies

to both the naked white-box DES implementation of Link and Neumann [9], and to the white-box DES implementation with external encodings as presented by Chow *et al.*. We have also implemented this cryptanalysis and successfully conducted several tests on binaries of white-box DES implementations.

The reminder of this paper is organised as follows: in Sect. 2 we give an overview of the white-box implementation of DES as proposed by Chow *et al.* [4]. The core of this paper, the cryptanalysis of the resulting obfuscated cipher, is described in Sect. 3. We have also implemented our attack and performed tests on white-box DES binaries. The results and considerations of the implementation are described in Sect. 4 and 5.

2 White-Box DES

For the sake of completeness and to synchronise with the terminology used in the description of the cryptanalysis, an overview of DES and white-box DES is presented in this section.

The *Data Encryption Standard* (DES) is a block cipher operating on 64-bit blocks with a key length of 56 bits; it has been adopted by NIST in 1977 as the encryption standard with the perspective of being fast, simple and secure. Although its successor AES (*Advanced Encryption Standard*) is designed to meet new security requirements and the evolution of computers, DES or variants like 3DES are still implemented and deployed in various applications, in particular in the financial sector.

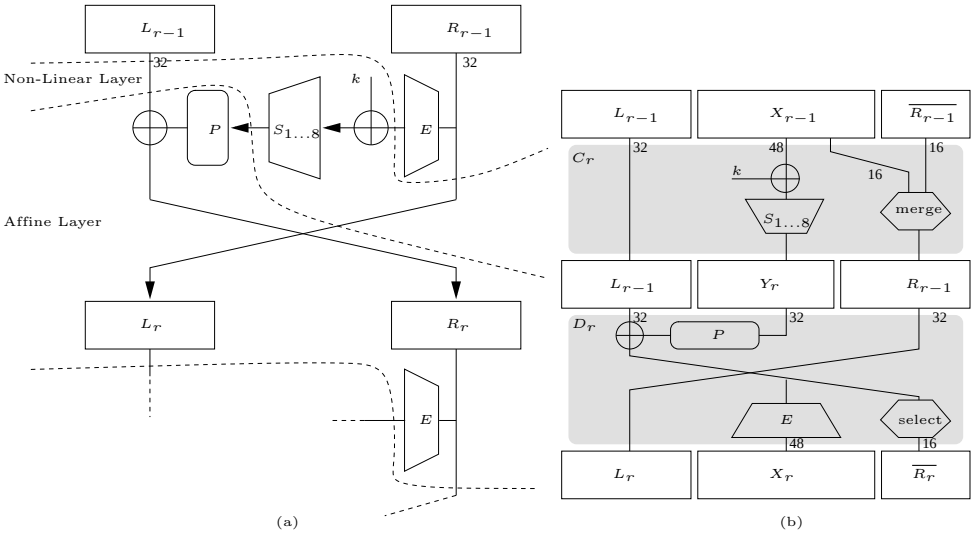

Fig. 1. (a) One round of DES (b) One round of white-box DES

DES is a Feistel cipher with 16 rounds, embedded between an initial permutation IP before the rounds, and its inverse permutation IP^{-1} after the last round. Fig. 1 (a) depicts one round of DES. It has the following building blocks: an expansion operation E; an addition of a 48-bit round key K^r which is generated from the key schedule; 8 S-box operations S_i (each S-box is a non-linear mapping from 6 bit to 4 bit); and a bit permutation P.

The main idea of compiling DES as a white-box implementation is to express DES as a sequence of lookup tables, and to obfuscate these tables by encoding their inputs and outputs. In this section, we discuss the steps for implementing DES with an embedded secret key, as described by Chow *et al.* [4] and improved by Link *et al.* [9] and Wyseur and Preneel [12].

As the S-boxes are the only non-linear component of the round function, we can divide the round into a non-linear layer, denoted by C_r, and a linear layer, denoted by D_r. This is depicted by the dotted lines in Fig. 1 (a), while the alternative representation is depicted in Fig. 1 (b).

2.1 T-boxes

The first layer of each white-box DES round consists of 12 T-boxes, which are defined as

$$\begin{cases} T_j^r = b_0 b_1 || b_2 b_7 || S_j(b_2 b_3 b_4 b_5 b_6 b_7 \oplus k_j^r) & (a)\ \forall j = 1 \dots 8 \\ T_j^r = b_0 b_1 b_2 b_3 || b_4 b_5 b_6 b_7 & (b)\ \forall j = 9 \dots 12 \end{cases}$$

Here r denotes the round number ($1 \leq r \leq 16$), $b_{0\dots7}$ represent the 8 input bits to each T-box, and k_j^r represents 6 bits of the round key. Fig. 2.1 depicts both types of T-boxes.

These 12 T-boxes represent the function C_r of round r of the DES implementation. The first 8 T-boxes are called *non-linear T-boxes*, as they contain the non-linear S-boxes. Furthermore, each of these 8 non-linear T-boxes passes on 2 bits of L_{r-1}, and the 2 outer input bits to the S-box. Due to the construction of the DES S-boxes, which have a bijective relation between the 4 middle input bits $b_3 b_4 b_5 b_6$ and the output bits, these T-boxes are bijective 8-to-8 bit bijections. This property is often referred to as having entropy 8. This design property is needed to prevent the T-boxes to leak information as described by Chow *et al.* [4]. Variations on the design [12] can be introduced, such as rearranging bypass bits, as long as the design properties are fulfilled.

The 8 non-linear T-boxes bypass 16 bits of L_{r-1} and 16 bits of R_{r-1} in total. The other 16 bits of L_{r-1} and 16 bits of R_{r-1} are bypassed using 4 linear T-boxes. We call these $T_j^r (\forall j = 9 \dots 12)$ *bypass T-boxes*, and the 16 bits of R_{r-1} they bypass we denote *restricted bits*, denoted by $\overline{R_{r-1}}$.

A first randomisation we apply is to shuffle the 12 T-boxes of C_r such that it is not straightforward to know which S-box is contained into which T-box. As shuffling is a linear operation: we can implement this operation in the affine operations D_{r-1} (shuffle) and D_r (unshuffle). Denote with π the shuffle operation which maps the T-box with internal S-box S_i to $T_{\pi(i)}$.

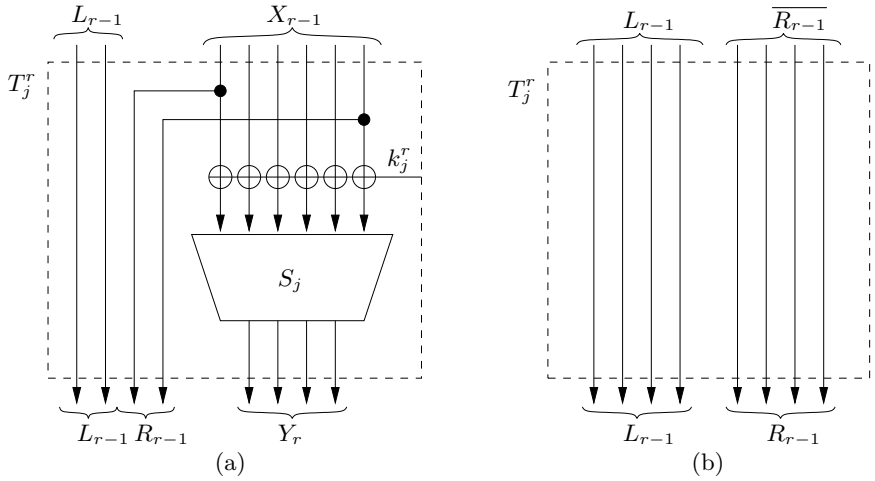

Fig. 2. The 2 types of T-boxes: (a) non-linear T-box with internal S-box (b) bypass T-box

2.2 Implementing the linear operations

The transformation D_r is a linear transformation of 96 bits to 96 bits, and can thus be implemented as a 96×96 bit matrix M multiplication $y = Mx$. In order to implement this multiplication as a network of lookup tables, we use a technique called matrix decomposition. The main strategy is to subdivide the 96×96 bit matrix M into $m \times n$ sub-matrices. Each sub-matrix $m \times n$ represents how an n-bit sub-vector of the input vector affects an m-bit sub-vector of the output vector. We implement each sub-matrix as a lookup table with 2^n rows of m bits.

In order to implement the full linear operation, the results of the outputs of the sub-matrix lookup tables must be XOR-ed. Each XOR operation applies to $2m$ bits and results into m bits. Putting it all together, implementing these XOR operations as a lookup table of 2^{2m} rows of m bits results into the full linear operation to be a network of lookup tables. Such a network is presented in Fig. 2.2.

Although lookup tables occupy much more space than linear operations, we need to implement the linear operations as lookup tables to apply non-linear encodings to the network. For both security and optimisation purposes, D_r is decomposed into 8×4 lookup tables ($m = 4, n = 8$). As every 8 bit T-box result of C_r is then fed into 24 decomposed tables of the affine operation D_r, we can fold this T-box with every of its following 8×4 lookup tables, and thus eliminate the 8×8 lookup tables, condensing the implementation. This optimisation step has been suggested by Link *et al.* [9].

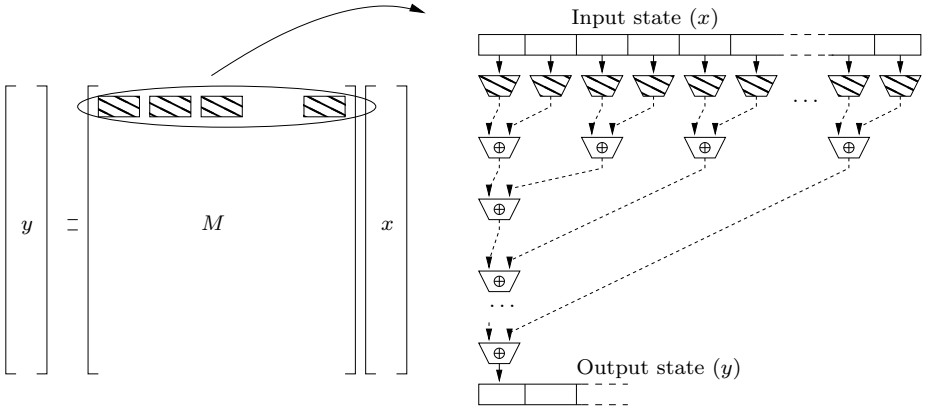

Fig. 3. Schematic overview how m bits of y can be computed from the x through a network of lookup tables.

2.3 Re-indexing and delinearisation

The steps mentioned above transform DES into a network of lookup tables with an embedded key. At this point, these lookup tables do not provide much security as it is not difficult to extract the key from the content of the tables. To protect this content, input/output encodings are applied to these lookup tables.

Let Λ be a lookup table, and f and g be random bijections. Then $g \circ \Lambda \circ f^{-1}$ is defined as its encoded version. We encode all lookup tables in the network, in such a way that the output encoding is cancelled by the input decoding incorporated into the next lookup table. Thus, consider a sequence of connected lookup tables

$$\Lambda_n \circ \Lambda_{n-1} \circ \ldots \circ \Lambda_2 \circ \Lambda_1 \,.$$

Then, in a simplified description, these lookup tables are encoded with random bijections f_i, g_i such that

$$(g_n \circ \Lambda_n \circ g_{n-1}^{-1}) \circ (g_{n-1} \circ \Lambda_{n-1} \circ g_{n-2}^{-1}) \circ \ldots \circ (g_2 \circ \Lambda_2 \circ g_1^{-1}) \circ (g_1 \circ \Lambda_1 \circ f_1)$$

with $f_i = g_{i-1}^{-1}$. This is a simplified description, because in practice, these encodings are the concatenation of small nibble encodings. As the lookup tables have an output of $m = 4$ bits, the encodings cannot have a larger dimension. Note that wider input and output encodings are not possible, as the delinearisation cannot exceed the boundaries of the lookup tables they are applied to. Applying all these encodings results into external encodings. Thus, a cipher is transformed into a cipher with external encodings:

$$E_k \rightarrow G \circ E_k \circ F.$$

Note that in the recommended variant as presented by Chow *et al.*, $F = M_1 \circ M_0$ and $G = M_4 \circ M_3$, where M_0 and M_4 are affine mixing bijections. A mixing

bijection is a bijective affine transformation which attempts to maximise the dependency of each output bit on all input bits. M_1 combines the initial DES permutation IP and the expansion E, while M_3 combines the final operations such as the DES permutation P and the inverse initial permutation IP^{-1}. Both F and G are encoded with nibble encodings in order to make them non-linear.

Without these external encodings, a white-box implementation (that is, a naked white-box implementation) becomes potentially vulnerable in the first and last round. Attacks on a naked implementation have been presented in [4,9,7].

3 Cryptanalysis

A detailed examination of the white-box DES implementation as presented by Chow *et al.* shows that, when encrypting a plaintext, the relation between the internal states does not behave randomly. With the internal state before round r, we denote the 96 bits that represent $L_{r-1}||X_{r-1}||\overline{R_{r-1}}$, the input to the T-boxes of round r, i.e., the 12-bytes vector $v_1^r||v_2^r||\dots v_{12}^r$, where v_j^r is the encoded input to an encoded T-box T_j^r. Each state can only be encoded with a concatenation of nibble encodings. Because wider input and output encodings are not feasible, and the DES design properties that input bits affect only a limited amount of output bits of the round function, we can observe that 8-bit chunks of the state to round r do not affect all 8-bit chunks of the state to round $r+1$.

This observation results into the main idea of our cryptanalysis: from the propagation of a difference to the input of an encoded T-box, we obtain information about the internal behaviour of this difference. This information is further refined using a repeated process of difference propagation observation, until we have exactly identified a set of differences to the input of an encoded T-box. From this set, we will be able to recover the DES key embedded into the implementation.

Below, we present the steps to identify differences to the input of T-boxes, and how this results into the recovery of the embedded secret key. In Sect. 3.1, we identify the set of differences which represent flips of $\overline{R_{r-1}}$ bits. This leads to the identification of flips of the two middle input bits of S-boxes in round $r+2$, and further into the identification of single input bit flips to S-boxes in round $r+3$, described in Sect. 3.2. This information is then used in Sect. 3.3 to identify the S-boxes contained inside the T-boxes, and the value of the input to these S-boxes. In Sect. 3.4, we explain how this results into the recovery of the embedded key.

Initialisation phase. During the initialisation of our cryptanalysis, we choose a random plaintext and run it through the implementation, storing all internal states. For this plaintext that we will deduce the corresponding inputs to the S-boxes. The value of the initial plaintext itself is of no importance, because the key recovery will only rely on the difference relation between the inputs to the S-boxes. Hence, this cryptanalysis also works for the white-box DES

implementation with external encodings, for which an encoded plaintext is fed into DES.

3.1 Finding restricted bit flips

Let T_j^r be an arbitrary encoded T-box in round r, encoded with input encoding f_j^r and output encoding g_j^r. Let v_j^r denote its 8-bit input vector set by the initialisation phase. In this section we present an algorithm to construct the set $\mathcal{S}_{\overline{R}}(T_j^r) = \{\Delta v = v_j^r \oplus v' | v' \in GF(2)^8; v' \neq v_j^r; f_j^r(\Delta v)$ an $\overline{R_{r-1}}$ bit flip$\}$ of all input differences to the encoded T_j^r which represent flips of *one* or *two* restricted bits ($|f_j^r(\Delta v)| \in \{1, 2\}$). Similarly, we define the sets $\mathcal{S}_R(T_j^r)$ and $\mathcal{S}_{R \setminus \overline{R}}(T_j^r)$. A difference Δv is applied to T-box T_j^r as follows: substitute the j^{th} byte of the internal state before round r from v_j^r into v', and compute the round function $D_r \circ C_r$ with this new internal state as input.

The algorithm consists of two parts: (1) constructing the set $\mathcal{S}_R(T_j^r)$ of all differences which represent *single* bit flips and some *double* bit flips of R_{r-1}, and (2) to divide this set into $\mathcal{S}_{\overline{R}}(T_j^r)$ and $\mathcal{S}_{R \setminus \overline{R}}(T_j^r)$.

Finding single R_{r-1} bit flips. Let $\Delta v : v_j^r \rightarrow v_j^r \oplus \Delta v$ be a difference for the input of T_j^r while the inputs v_l^r to the other T-boxes T_l^r are set to the values from the initialisation phase ($\forall l \neq j : \Delta v_l^r = 0$). We observe the difference propagation in the next rounds, and formulate two observations:

Observation 1 *If Δv represents a single bit flip of R_{r-1}, then in round $r + 2$, at most 2 T-boxes are affected (i.e., its input changes).*

Proof. When Δv represents a single R_{r-1} bit flip, then in round $r+1$ it represents a single L_r bit flip, as the reader can deduce from Fig. 1(b). Because of the expansion and selection operation, this will result into 2 bits flipped to round $r+2$ (one of X_{r+1} and one of $\overline{R_{r+1}}$; or both X_{r+1} flips). Thus at most 2 T-boxes in round $r + 2$ are affected. □

Observation 2 *If Δv represents flips of L_{r-1} or Y_r bits, then there is a large probability that more than 2 T-boxes are affected in round $r + 2$.*

If not more than 2 T-boxes are affected, we denote Δv as a *false positive*. In this case, the input to an S-box in round $r+1$ changes in at least one and at most 3 bits. The effect on the output bits of this S-box depends on its other input bits, which depend on the inputs v_l^r set at the initialisation phase. Therefore, the number of affected T-boxes in round $r + 2$ will very likely change if we set other inputs to T_l^r.

Thus, if the number of affected T-boxes in round $r + 2$ is larger than 2, Δv does not represent a flip of R_{r-1} bits (see Observation 1). If the number is smaller than or equal to two, then we need to perform extra checks with other input bits to remove false positives. With each extra check, the probability of having a false positive reduces. In the implementation as presented by Chow *et al.*, the total

differences representing R_{r-1} bit flips for all the T-boxes of one round, is exactly 40: 16 *single* R_{r-1} bit flips originating from X_{r-1}, 16 *single* $\overline{R_{r-1}}$ bit flips, and 8 *double* $\overline{R_{r-1}}$ bit flips. If $\sum_j |\mathcal{S}_R(T_j^r)| > 40$, then extra checks are required to remove the false positives. Note that we cannot distinguish some double flips of $\overline{R_{r-1}}$ bit flips from single bit flips. For the implementation of Chow *et al.*, this is the case when the bypass bits of both middle bits of an S-box are flipped. Thus, 8 double flips are detected for one round.

Algorithm 1 describes this procedure, where all input differences Δv of T_j^r representing these flips are stored in the set $\mathcal{S}_R(T_j^r)$.

Algorithm 1 Selecting single R_{r-1} bit flips

```
1: Set all v_l^r
2: for all Δv ∈ GF(2)^8\{0} do
3:    Compute 2 round functions
4:    if # affected T-boxes ≤ 2 then
5:       extra check: set new v_l^r;   ∀l ≠ j
6:       Compute 2 round functions
7:       if # affected T-boxes ≤ 2 then
8:          Δv → S_R(T_j^r)
9:       end if
10:   end if
11: end for
```

Split R_{r-1} into $\overline{R_{r-1}}$ and $R_{r-1}\backslash\overline{R_{r-1}}$ flips. Let Δv represent flips of R_{r-1} bits.

Observation 3 *If $\Delta v \in \mathcal{S}_R(T_j^r)$ represent a flip of $\overline{R_{r-1}}$ bits, there are exactly 2 propagated differences in round $r+2$. One difference will affect more than 2 T-boxes in round $r+4$, while the other difference will affect at most 2 T-boxes in round $r+4$.*

Proof. Let $\Delta v \in \mathcal{S}_R(T_j^r)$ represent a (single or a double) flip of $\overline{R_{r-1}}$ bits. Then, in round $r+2$, this will propagate to a flip of the two middle input bits of an S-box S_m in T-box T_m^{r+2}. Denote Δm the propagated input difference to T_m^{r+2}. Furthermore, this flip will also be bypassed because of the selection operation (see Fig. 1(b)). If this would be bypassed by T_m^{r+2} as well, then this T-box has entropy 7, in contradiction to the T-box design. Thus a second T-box T_n^{r+2} is affected, with input difference Δn. Therefor, Δv will affect exactly 2 T-boxes T_m^{r+2}, T_n^{r+2} with input differences $\Delta m, \Delta n$.

Consider the following DES S-box design properties [5]:

$$\Delta_{in} = 0wxyz0 \Rightarrow |\Delta_{out}| \geq 2 \tag{1}$$

$$|\Delta_{in}| = 1 \Rightarrow |\Delta_{out}| \geq 2 \tag{2}$$

with $wxyz \in GF(2)^4 \backslash \{0\}$. Because of (1), Δm represents a flip of at least two Y_{r+2} bits at the output of the S-box. Due to the DES permutation P diffusion property and (2), Δm will affect more than 2 T-boxes in round $r+4$. Δn represents a flip of bits of R_{r+1}, and affect no more than two T-boxes in round $r+4$ (see Observation 1). □

Observation 4 *If $\Delta v \in \mathcal{S}_R(T_j^r)$ represents a flip of $R_{r-1} \backslash \overline{R_{r-1}}$ bits, then two propagated differences in round $r+2$ will affect more than two T-boxes in round $r+4$.*

Proof. If $\Delta v \in \mathcal{S}_R(T_j^r)$ represents a single flip of $R_{r-1} \backslash \overline{R_{r-1}}$ bits, then for 2 S-boxes in round $r+2$, exactly one bit will be affected. Because at most one S-box can be contained in a T-box, exactly two T-boxes will be affected. Because of (2), both will represent a flip of at least two Y_{r+2} bits. Because of the DES permutation P diffusion property, both propagated differences in round $r+2$ will then affect strictly more than two T-boxes in round $r+4$. □

Hence, we also have a tool to distinguish non-linear T-boxes. From Observation 3, we deduce that T-box T_m^{r+2} is a non-linear T-box as it contains the S-box S_m, while both affected T-boxes of Observation 4 are non-linear T-boxes as well. In Algorithm 2, the splitting procedure using the latter two observations is described. Note that during the algorithm, we also store the differences Δm representing flips of middle bits $(b_4 b_5)$ to an S-box S_m in the set $\mathcal{S}_M(T_m^{r+2})$. Note also that from Observation 3 and Observation 4, Observation 1 can be defined more strictly to require *exactly* 2 T-boxes to be affected.

Algorithm 2 Split R_{r-1} into $\overline{R_{r-1}}$ and $R_{r-1} \backslash \overline{R_{r-1}}$ flips

for all $\Delta v \in \mathcal{S}_R(T_j^r)$ **do**
  Compute 2 round functions
  $\Delta m, \Delta n \leftarrow$ propagated differences in round $r+2$ of $T_m^{r+2}, T_n^{r+2} \quad m \neq n$
  $\delta m \leftarrow$ # affected T-boxes in round $r+4$ propagated by Δl in round $r+2$.
  $\delta n \leftarrow$ # affected T-boxes in round $r+4$ propagated by Δm in round $r+2$.
  if $\delta m > 2$ and $\delta n = 2$ **then**
    $\Delta v \rightarrow \mathcal{S}_{\overline{R}}(T_j^r); \Delta m \rightarrow \mathcal{S}_M(T_m^{r+2})$
    Denote T_m^{r+2} as non-linear T-box
  else if $\delta m = 2$ and $\delta n > 2$ **then**
    $\Delta v \rightarrow \mathcal{S}_{\overline{R}}(T_j^r); \Delta n \rightarrow \mathcal{S}_M(T_n^{r+2})$
    Denote T_n^{r+2} as non-linear T-box
  else if $\delta m > 2$ and $\delta n > 2$ **then**
    $\Delta v \rightarrow \mathcal{S}_{R \backslash \overline{R}}(T_j^r)$
    Denote both T_m^{r+2} and T_n^{r+2} as non-linear T-box
  end if
end for

The combination of Algorithm 1 and Algorithm 2 results into the following useful information:

$$\begin{cases} \mathcal{S}_{\overline{R}}^{r} = \cup_j \mathcal{S}_{\overline{R}}(T_j^r) \text{ Differences representing restricted bit flips} \\ \mathcal{S}_{M}^{r+2} = \cup_j \mathcal{S}_{M}(T_j^{r+2}) \text{ Differences representing S-box middle bit flips} \\ T_{\pi(1)}^{r+2} \dots T_{\pi(8)}^{r+2} \text{ The 8 non-linear T-boxes } (\pi \text{ unkown}) \end{cases}$$

Note that, using the set $\mathcal{S}_{\overline{R}}^{r} = \cup_j \mathcal{S}_{\overline{R}}(T_j^r)$, we can efficiently compute the set $\mathcal{S}_{\overline{R}}^{r+2}$ for round $r+2$ without using Algorithm 1. As described in Observation 3, from $\Delta v \in \mathcal{S}_{\overline{R}}^{r}$, we compute $\Delta n \in \mathcal{S}_{\overline{R}}^{r+2}$. Because of the one-to-one relation between Δv and Δn; $|\mathcal{S}_{\overline{R}}^{r+2}| = |\mathcal{S}_{\overline{R}}^{r}| = 24$. We can conclude that we have all the required differences for the set $\mathcal{S}_{\overline{R}}^{r+2}$. Thus, when for two consecutive rounds, $\mathcal{S}_{\overline{R}}$ is found, we can compute this set for all subsequent rounds using Observation 3 only.

Complexity. We define the complexity of the cryptanalysis as the number of round functions of the white-box implementation that need to be computed. The step described in this section has the highest complexity of our cryptanalysis. Because of the lack of any prior information on internal flips, all differences have to be computed through several rounds in order to gain this bit flip information. In Algorithm 1, for all 12 T-boxes, and all $2^8 - 1$ possible differences, the difference propagation needs to be computed through 2 rounds, which corresponds to a total of $12 \cdot (2^8 - 1) \cdot 2 = 6120$ round function computations. With probability ε, besides the 40 expected differences, λ false positives occur. Therefor, $\varepsilon \cdot (40 + \lambda)$ extra checks need to be computed, with $\varepsilon < 1$ and $\lambda = \mathcal{O}(1)$. Algorithm 2 requires for each difference of $\mathcal{S}_R$ 6 round computations (2 for Δv, 2 for Δl and 2 for Δm). Thus 240 round functions need to be computed. The reduced version which only uses Observation 3 as mentioned above, only requires 144 round computations (6 round computations for each of the 24 differences of $\overline{R}$).

In total, both algorithms together have complexity $12 \cdot (2^8 - 1) \cdot 2 + \varepsilon \cdot (40 + \lambda) \cdot 2 \leq 2^{13}$ for one round r. When applying this for two consecutive rounds, only 144 round function computations are required for each subsequent round. Hence, the complexity of the algorithm to compute all the required information for all the possible rounds is approximately 2^{14}.

3.2 Finding single bit flips

In Sect. 3.1, differences representing flips of the 2 middle bits ($b_4 b_5$) of the S-boxes of round $r+2$ are found. Let T_j^{r+2} be an arbitrary non-linear T-box in round $r+2$, and $\mathcal{S}_M(T_j^{r+2})$ its set of middle bit flips. We have $\mathcal{S}_M(T_j^{r+2}) = \{\Delta m_1, \Delta m_2, \Delta m_3\}$ with $\Delta m_i : v_j^{r+2} \rightarrow v_j \oplus \Delta m_i$ the 3 generated differences. One can verify that, with the exception of S-box 8, each of the four output bits of the S-box S_j^{r+2} are flipped at least once by going through one of the values $v_j^{r+2} \oplus \Delta m_1, v_j^{r+2} \oplus \Delta m_2, v_j^{r+2} \oplus \Delta m_3$. Furthermore, as the middle bits are not bypassed in the same T-box, no other output bits of the T-box are affected. Due to the diffusion property of the DES permutation P, each of the four output bits affects a different S-box in round $r+3$ [5]. Thus, the propagated differences to

the T-boxes of round $r+3$ caused by a difference of middle bit flips in round $r+2$ represent single bit flips.

This property is only violated for S-box 8. For the input $11b_4b_501$, with b_4 and b_5 any possible bit, we cannot flip the rightmost output bit, which affects S-box 5 and 6 in round $r+3$. Thus, with a probability of 1/16, we do not find all single bit flips of round $r+3$. In this case, we can start the cryptanalysis all over with another initial plaintext in order to find all single bit flips of round $r+3$. However, it will become clear in the next section, that it does not harm not to have all information. This issue will thus not affect our cryptanalysis.

Algorithm 3 Finding single bit flips

1: **for all** $\Delta v \in \mathcal{S}_M(T^{r+2}_{\pi(j)}) \quad j = 1 \dots 8$ (for non-linear T-boxes) **do**
2: Compute one round function
3: **for all** Δw_i propagated difference to a T-box T_i^{r+3} **do**
4: $\Delta w_i \rightarrow \mathcal{S}_S(T_i^{r+3})$
5: **end for**
6: **end for**

Complexity. The complexity of this step, described in Algorithm 3 is negligible as for each $\Delta m \in \mathcal{S}^r_M$, one round function needs to be computed. Hence, for each round, the complexity of this algorithm is 24.

3.3 Obtaining the inputs to the S-boxes

Let T_j^{r+3} be an arbitrary non-linear T-box in round $r+3$. Using the acquired information from the steps above, we deploy a filter algorithm to identify the S-box $(S_{\pi^{-1}(j)})$ in the T-box T_j^{r+3}, and find the value of its 6 input bits $(f_j^{r+3}|_{2\dots7}(v_j^r) \oplus k_j^{r+3})$.

We define the set $\mathcal{P}(T_j^{r+3}) = \{(S_q, w_l) | 1 \leq q \leq 8, w_l \in GF(2)^6\}$ of all possible pairs of S-boxes and input vectors. Our goal is to reduce this set by comparison of the number of affected T-boxes in round $r+4$ when a difference $\Delta v_i \in \mathcal{S}_S(T_j^{r+3}) \cup \mathcal{S}_M(T_j^{r+3})$ is applied to the input of T_j^{r+3}, and the number of affected S-boxes in a non-white-box DES simulation with a pair $(S_q, w_l) \in \mathcal{P}(T_j^{r+3})$. We denote δ_i the number of affected T-boxes when Δv_i is applied, and δ'_i the number of affected S-boxes in the simulation process with Δw_i, where $\Delta w_i : w_l \rightarrow w_l \oplus f_j^{r+3}|_{2\dots7}(\Delta v_i)$ is the flip of the input bits to S-box S_q represented by Δv_i.

If (S_q, w_l) is valid pair, it should satisfy the following conditions:

- There can only be one S_q for each round.
- $\Delta v_7 = \mathcal{S}_M(T_j^{r+3}) \backslash \mathcal{S}_S(T_j^{r+3})$ is the flip of both middle bits, represented as $\Delta w_7 = 001100$, for which δ'_7 can be computed. Because Δv_7 only affects bits of Y_{r+3}, δ_7 must be equal to δ'_7.

- $\{\Delta v_3, \Delta v_4\} = \mathcal{S}_M(T_j^{r+3}) \cap \mathcal{S}_S(T_j^{r+3})$ represent the two single flips of the input bits to the S-box, but we do not know in which order. Moreover they only affect bits of Y_{r+3}, and thus we must have $\{\delta'_3, \delta'_4\} = \{\delta_3, \delta_4\}$.
- Similarly $\{\delta'_2, \delta'_5\} \in \{\delta_1, \delta_2, \delta_5, \delta_6\}$.
- The differences affecting the outer input bits affect bits of R_{r+2}, and therefore the number of affected S-boxes can be smaller than the number of affected T-boxes, which should be taken into account when comparing $\{\delta'_1, \delta'_6\}$ to $\{\delta_1, \delta_2, \delta_5, \delta_6\} \backslash \{\delta'_2, \delta'_5\}$.

Any pair (S_q, w_l) that does not fulfil these conditions is removed from the set $\mathcal{P}(T_j^{r+3})$. If only pairs with one type S_q remain, then this S_q is the internal S-box of $T_j^{r+3}(\pi(q) = j)$. Finding an internal S-box results into an avalanche effect in finding solutions for other non-linear T-boxes, because S-boxes are unique in the same round (first condition), and because of the relation between S-boxes of consecutive rounds. E.g., S_1 in round r does not affect S-box S_1 and S_7 in round $r+1$. Moreover, if for example S_3 is identified in round $r+1$, then S_1 affects its second input bit, which allows us to narrow the conditions $(\delta_2 = \delta'_2)$.

As a result, the set $\mathcal{P}(T_j^{r+3})$ is reduced to a singleton (S_q, w_l), where $S_q = S_{\pi^{-1}(j)}$ is the internal S-box and $w_l = f_j^{r+3}|_{2\ldots7}(v_j^{r+3})$ the 6-bit input vector to this S-box.

Complexity. For each non-linear T-box, and each of its 7 differences, one round function is computed. The simulation process for each T-box needs to be performed at most $2^6 \cdot 8 = 2^9 (= |\mathcal{P}(T_j^r)|)$ times. Each simulation requires the computation of one single lookup table (S-box), while a white-box round requires $24 \cdot (12 + 11) = 552 \sim 2^9$ lookup table computations. Thus for each T-box, we can express the simulation complexity as one round function computation. The total complexity to compute the inputs to all S-boxes of one round is thus $8 \cdot (7 + 1) = 2^6$.

3.4 Key recovery

Given that we have found a sufficient amount of inputs to S-boxes, we can retrieve the embedded secret key using the following two different approaches:

- From the data expansion operation E, we know that prior to the key addition, the rightmost input bit of S_i^r equals the second leftmost bit of S_{i+1}^r, while the left most bit of S_{i+1}^r equals the second rightmost bit of S_i^r. This results into two independent equations, each with 4 variables (two input bits and their respective two key bits). Knowing the input bits from the algorithm described in Sect. 3.3, and one key bit, we are able to compute the other key bit.
- A second set of equations is built by following one bit i of R_r through several rounds. In round r, after the expansion operation E and round key bit k_j^r addition, this bit is used as an input bit to an S-box S_j^r. In round $r+1$, this

bit is XOR-ed with the value of bit $P^{-1}(i)$, which is the output of an S-box S_l. In round $r+2$, after the expansion operation and a round key bit k_j^{r+1} addition, this bit is used as an input bit to an S-box S_j. This results into an equation with three bits from S-boxes, and two round key bits. If all the bits related to the S-boxes are known, and one key bit is known, we can compute the other key bit.

Iterated use of these algorithms generates the DES key bits. When a new round key bit is computed, we can pull this back through the DES key schedule. This is possible, because the 48 bit round key is a fixed permutation of a subset of the 56-bit DES key. New key bits in turn result into new round key bits, for which the two described methods can be deployed.

The iterated recovery of key bits is initiated by guessing one single key bit. As a result, two complementary keys k_0 and k_1 are computed. Using the complementation property DES exhibits, we can show both keys are a valid solution. The complementation property of DES [11] is defined as

$$DES_k = \bigoplus_1 \circ DES_{k\oplus 1} \circ \bigoplus_1,$$

where $\bigoplus_1$ represents the XOR with the all one vector. Then

$$\begin{aligned} G \circ DES_k \circ F &= G \circ \bigoplus_1 \circ DES_{k\oplus 1} \circ \bigoplus_1 \circ F \\ G \circ DES_k \circ F &= G' \circ DES_{k\oplus 1} \circ F' \,. \end{aligned}$$

If $F = M_1 \circ M_0$ and $G = M_4 \circ M_3$, we can define $F' = (\bigoplus_1 \circ M_1) \circ M_0$ and $G' = M_4 \circ (M_3 \circ \bigoplus_1)$. Thus, when k is the original DES key, and F, G the external encodings used to build the white-box DES implementation, then the complementary key $k \oplus 1$ is also a valid DES key with external encodings F', G'.

4 Implementation

We have implemented our cryptanalysis in C++, and conducted tests on a Pentium M 2GHz. On average, about $6000 \leq 2^{13}$ obfuscated round functions of the white-box DES implementation needed to be computed to check the difference propagations. This is less than our complexity study indicated, due to some extra optimisations we have applied. Finding bit flips of R_{r-1} (Algorithm 1) corresponds to the largest part of the complexity. Introducing criteria for round $r+1$ in Observation 1 substantially improves the algorithm efficiency. Moreover, we only need 8 consecutive obfuscated round functions for the attack to succeed. There is no restriction which window of 8 round functions to chose.

The space complexity is negligible, as most space is used in Sect. 3.3 to store the set $\mathcal{P}(T_j^r)$ of candidate pairs (S_i, w_l). We can also choose to store the simulations of these pairs. They can be pre-computed because simulation does not require any information or the implementation or the key.

In the conducted tests on several white-box DES implementations, our cryptanalysis algorithm extracted the DES key in 0.64 seconds. Due to the time required to generate a lot of random sequences for the random delinearisation process, the time to generate a white-box DES implementation is comparable to the time to extract the secret key from the implementation.

5 Conclusion

We have described how the embedded secret key of an encoded white-box DES implementation can be extracted. This cryptanalysis applies to the implementation as presented by Link and Neumann [9], an improvement of the proposed implementation of Chow *et al.* [4] which includes external encodings. We apply a differential cryptanalysis on the obfuscated rounds. The external encodings are not incorporated in the attack, hence our attack is independent from the definition of these encodings, in contrast to the attack of Goubin *et al.* [6]. The downside is that we did not present how to recover these external encodings as we concentrate on the recovery of the secret key. However, reconstructing the encodings is rather straightforward.

The success of this cryptanalysis originates from properties which are specific to DES. The confusion property of the DES S-boxes, the diffusion property of the DES permutation P and the design of the expansion operation are the foundations to extract useful information. As a consequence, difference propagation results into key extraction. The general conclusion leads to a bootstrap problem: precisely what makes the cipher strong against attacks in the black-box model, is what makes it weak in the white-box model. But, when designing a cipher which can securely be implemented as a white-box implementation, protection against black-box attacks needs to be incorporated as well, as attacks in a black-box model are also possible in a white-box model. This indicates the challenge of further research in white-box cryptography.

Acknowledgements

This work has been funded in part by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen); the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, the Research Foundation - Flanders (FWO-Vlaanderen); and the Belgian Fundamental Research Network on Cryptology and Information Security (IUAP - BCRYPT).

References

1. Eli Biham and Adi Shamir. Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer (extended abstract). *Lecture Notes in Computer Science*, 576, 1991.

2. Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box AES implementation. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2004.
3. Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-Box Cryptography and an AES Implementation. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2002.
4. Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A white-box DES implementation for drm applications. In *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002.
5. D. Coppersmith. The data encryption standard (DES) and its strength against attacks. *IBM J. Res. Dev.*, 38(3):243–250, 1994.
6. Louis Goubin, Jean-Michel Masereel, and Michael Quisquater. Cryptanalysis of white box DES implementations. Cryptology ePrint Archive, Report 2007/035, 2007. `http://eprint.iacr.org/`.
7. Matthias Jacob, Dan Boneh, and Edward W. Felten. Attacking an obfuscated cipher by injecting faults. In *Digital Rights Management Workshop*, pages 16–31, 2002.
8. Tim Kerins and Klaus Kursawe. A cautionary note on weak implementations of block ciphers. In *1st Benelux Workshop on Information and System Security (WISSec 2006)*, page 12, Antwerp,BE, 2006.
9. Hamilton E. Link and William D. Neumann. Clarifying obfuscation: Improving the security of white-box DES. In *ITCC (1)*, pages 679–684, 2005.
10. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.
11. Charles P. Pfleeger. *Security in computing.* Prentice-Hall, Englewood Cliffs, New-Jersey, 1989.
12. Brecht Wyseur and Bart Preneel. Condensed white-box implementations. In *Proceedings of the 26th Symposium on Information Theory in the Benelux*, pages 296–301, Brussels,Belgium, 2005. Werkgemeenschap voor Informatie- en Communicatietheorie.