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

Abstract. At DRM 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 its execution 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 [10]. In this paper, we present an algorithm to extract the secret key from such white-box DES implementations. The cryptanalysis is a differential attack on obfuscated rounds, and works regardless of the shielding external encodings that are applied. The cryptanalysis has a average time complexity of 2^{14} and a negligible space complexity.

Keywords: White-Box Cryptography, Obfuscation, DES, Data Encryption Standard, Cryptanalysis.

1 Introduction

White-box cryptography aims to protect secret keys by embedding them into a 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 has full control over the implementation and its execution environment. This includes static and dynamic analysis of the implementation, altering of computation, and modification of internal variables. In such a model, it is much more difficult to protect cryptographic implementations than in the classical black-box model. In the black-box model, an adversary can only use the input and output behaviour of the implementation in order to find the key. Another model is the grey-box model, where an adversary can use side-channel information such as power consumption, and timing information.

C. Adams, A. Miri, and M. Wiener (Eds.): SAC 2007, LNCS 4876, pp. 264–277, 2007.

For the black-box model, several cryptographic block ciphers have been proposed, such as DES *(Data Encryption Standard)* [13], and its successor AES *(Advanced Encryption Standard)*. Although these ciphers provide cryptographic strength in their full number of rounds, attacks have been presented on reduced round versions. Cipher designers aim to reduce the number of rounds, for which a cipher provides sufficient security, while cryptanalysists try to construct an attack on as many rounds as possible. For AES-128 and AES-192, a cryptanalysis on 7 and 8 rounds has been presented respectively (out of 10 and 12 rounds) [9]. 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 intends to preclude cryptographic attacks on a reduced number of rounds, timing attacks, such as cache attacks (e.g., [11]) or implementation attacks [8]. Parallel with the white-box DES implementation proposal, Chow *et al.* [3] described a white-box AES implementation based on similar design principles. For both implementations, a variant was presented that is shielded with external encodings. Several publications describe cryptanalysis results of 'naked' white-box DES implementations, i.e., without the shielding external encodings [4,7,10]. The encoded white-box AES implementation has been cryptanalysed by Billet *et al.* [2]. They use algebraic properties of the AES to attack the implementation on the obfuscated round functions.

In this paper, we describe a cryptanalysis which applies to both naked and encoded white-box DES implementations. Independently, Goubin *et al.* [6] present similar results. Their paper describes a cryptanalysis of the improved naked white-box DES implementation proposed by Link and Neumann [10]. Based on this attack and the analysis of the typical external encodings, an attack is derived for encoded white-box DES implementations. In contrast, the attack we discuss in this paper is *independent* of the definition of the external encodings. Hence, unlike the attack of Goubin *et al.*, a white-box DES implementation cannot be protected against our attack by choosing different 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. Because the attack is independent of the definition and implementation of the external encodings, it applies to both the (improved) naked and the encoded white-box DES implementations.

The reminder of this paper is organised as follows: in Sect. 2 we give a brief overview of the white-box DES implementation. The core of this paper, the cryptanalysis of the implementation, 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. Section 5 presents the conclusions.

2 White-Box DES Implementations

For the sake of completeness and to introduce the notation and terminology used in the description of the cryptanalysis, we briefly outline the construction of white-box DES implementations as presented by Chow *et al.* [4].

The *Data Encryption Standard* (DES) is a block cipher operating on 64-bit blocks and with a key length of 56 bits; it is a Feistel cipher with 16 rounds, embedded between an initial permutation IP before the first round, and its inverse permutation IP^{-1} after the last round. Fig. 1 (a) depicts one round of the 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 bits to 4 bits); and a bit permutation P.

A DES white-box implementation represents DES as a functional equivalent sequence of obfuscated lookup tables. In this section, we describe the transformation techniques as presented by Chow *et al.* [4]. Figure 1 (a) depicts one round of DES, and (b) a functionally equivalent representation which consist of the functions C_r and D_r. The DES permutation, XOR, and expansion operation are implemented as a 96-to-96 bit affine function D_r, which can be represented as a matrix. Using a technique referred to as *matrix decomposition* by Chow *et al.*, D_r is transformed into a sequence of lookup tables. To avoid sparse submatrices, D_r can be split into non-sparse matrices by introducing mixing bijections [4].

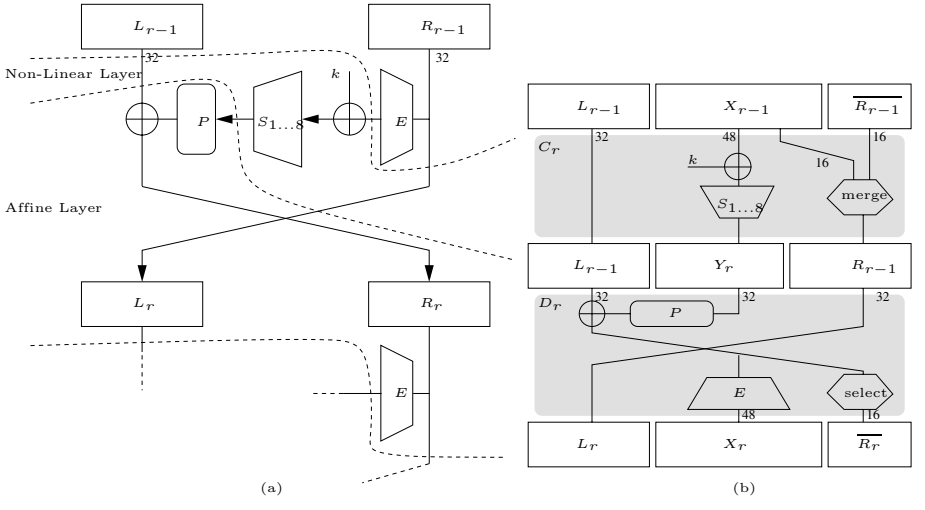

Fig. 1. (a) One round of DES (b) One round of white-box DES, before internal encodings are applied

At the heart of each round of the white-box DES implementation are 12 T-boxes that implement the C_r function. These T-boxes contain the S-boxes and the round key addition and are defined as

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

where 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. The first 8 T-boxes are called *non-linear T-boxes*, as they contain the non-linear S-boxes. The other 4 are called *bypass T-boxes*. The 12 T-boxes of C_r are defined in such a way that they are functionally equivalent to the round key addition, S-box operations and the bypass of all 32 left bits (L_{r-1}) and 32 right bits (R_{r-1}). Moreover, due to the bijective relation between the inner 4 input bits and the output bits of an S-box, these T-boxes are 8-to-8 bit bijections. This 8 bit entropy property is desirable as it prevents the isolated T-boxes to leak information as described by Chow *et al.* [4]. The order of the T-boxes can be permuted. Note that in that case, the affine operations D_{r-1} and D_r must be adjusted accordingly. Denote with π the permutation operation, i.e., S_i is implemented inside $T^r_{\pi(i)}$.

The result is a network of key dependent lookup tables. To protect the key information in these tables, input and output encodings are applied to them. Let Λ be a lookup table, and f and g be random bijections. Then $g \circ \Lambda \circ f^{-1}$ is defined an encoded lookup table. We encode all the lookup tables in the network in such a way that an output encoding is canceled by the input decoding incorporated into the next lookup table. Note that these input and output encodings are not wide, because they cannot exceed the boundaries of the lookup tables they are applied to. From now on, we refer to an encoded T-box as $g_i \circ T^r_{\pi(i)} \circ f_i^{-1}$, and the internal state as the 12-byte vector $f_1 || f_2 || \dots || f_{12}(\Phi_r(L_{r-1} || X_{r-1} || R_{r-1}))$ where Φ_r is the function which arranges the bits to the inputs of the T-boxes. Remark that in Fig. 1 the internal states are depicted unencoded.

Once the full network of lookup tables has been encoded, the input encodings at the beginning and output encodings at the end of the implementation are not canceled out. Without these encodings, we call the white-box DES implementation *naked*. Attacks on a naked implementation have been presented in [4,7,10]. In order to avoid such attacks, Chow *et al.* recommend to add affine mixing bijections before and after DES. As a result, not DES, but an encoded variant $G \circ DES_k \circ F$ is implemented. F and G are called *external encodings*.

3 Cryptanalysis

Examination of the white-box DES implementation as presented by Chow *et al.* shows that plaintext input differences between the rounds do not propagate randomly. Denote the internal state before round r as the 96 bits that represent the encoded version of $L_{r-1} || X_{r-1} || \overline{R_{r-1}}$. This is a 12-byte vector $v_1^r || v_2^r || \dots v_{12}^r$, where v_j^r is the encoded input to a T-box T_j^r. In our cryptanalysis we apply changes to the internal states and analyse their propagation in the consecutive

rounds to gain information about the implementation. This information is then used to recover the key. The applied technique builds on a generic strategy described for the first time by Billet *et al.* [2]. The cryptanalysis also applies to the improved implementation as presented by Link and Neumann [10], because only the inputs to the T-boxes are used. Merging C_r and D_r, or any change to $D_r \circ C_r$ (e.g., with mixing bijections) that does not change the inputs to C_r beyond the input size boundary, does not affect the attack.

Below, we present the steps to classify differences to the input of the T-boxes and show how this results in the recovery of the embedded secret key. In Sect. 3.1, we identify the set of differences which represent flips of restricted ($\overline{R_{r-1}}$) bits. This leads to the identification of flips of the two middle input bits of S-boxes in round $r+2$, and results also in the identification of single input bit flips to S-boxes in round $r+3$, as 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 precise value of the input to these S-boxes. In Sect. 3.4, we explain how this leads to the recovery of the embedded key.

Initialisation Phase. At the initialisation of our cryptanalysis, we choose a random plaintext and run it through the implementation, storing all internal states. We will deduce the inputs to the S-boxes for this plaintext in Sect. 3.3. Because we are only interested in the propagation of differences applied to the internal states, the value of the plaintext is of no importance. Hence, preceding external input encodings do not affect the success of this cryptanalysis.

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 the 8-bit input vector to the encoded T-box computed at the initialisation phase. In this section we present an algorithm to construct the set $\mathcal{S}_{\overline{R}}(T_j^r) = \{\Delta v = v_j^r \oplus v' \mid v' \in GF(2)^8; v' \neq v_j^r; f_j^r(v_j^r) \oplus f_j^r(v')$ 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 ($|f_j^r(v_j^r) \oplus f_j^r(v')| = 1, 2$). Similarly, we define the sets $\mathcal{S}_R(T_j^r)$ and $\mathcal{S}_{R \setminus \overline{R}}(T_j^r)$. An input difference Δv is applied to T-box T_j^r as follows: change 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 $\Delta v = v_j^r \oplus v' (= \Delta v_j^r)$ be a difference of the input of T_j^r while the inputs v_l^r to the other T-boxes T_l^r are fixed to the values from the initialisation phase ($\forall l \neq j : \Delta v_l^r = 0$). The following two properties can be proved for Δv. The proofs are given in Appendix A.

Property 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., their input change).*

Property 2. *If Δv represents flips of bits of L_{r-1} or Y_r, then in almost all cases more than 2 T-boxes are affected in round $r+2$. The exceptions (false positives) can be detected by repeating this process up to α times with different fixed inputs to the other T-boxes T_l^r.*

Hence, we are able to distinguish flips Δv that represent flips of R_{r-1} bits, and build the set $\mathcal{S}_R(T_j^r)$. Algorithm 1 describes this procedure. The total number of differences representing flips of bits of R_{r-1} for all the T-boxes of one round, is exactly 40: 16 *single* flips of bits of R_{r-1} originating from X_{r-1}, 16 *single* flips of bits of $\overline{R_{r-1}}$, and 8 *double* flips of bits of $\overline{R_{r-1}}$. To agree with property 2, these double flips of restricted bits are those that affect the both middle bits of an S-box in round $r+2$, and are bypassed together through the implementation. Therefore they cannot be distinguished from single bit flips of R_{r-1}. Because there are only 8 S-boxes, there cannot be more than 8 double bit flips. Depending on the design of Φ_r, the number of double flips can reduce, but this does not influence our cryptanalysis. To keep the discussion clear, we assume the bypass bits are ordered, and therefore we will have 8 double bit flips.

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:    while # affected T-boxes ≤ 2 and # checks ≤ α  do
5:       Extra check: set new v_l^r;   ∀l ≠ j
6:       Compute 2 round functions
7:    end while
8:    if # affected T-boxes ≤ 2 then
9:       Δv → S_R(T_j^r)
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. The following properties can be proved for $\Delta v \in \mathcal{S}_R(T_j^r)$. The proofs are given in Appendix A.

Property 3. *If Δv represents a flip of bits of $\overline{R_{r-1}}$, there are exactly 2 propagated differences in round $r+2$: Δm, Δn. One (say Δm, input difference to T-box T_m^{r+2}) will affect strictly more than 2 T-boxes in round $r+4$, the other difference will affect at most 2 T-boxes in round $r+4$. Moreover, T_m^{r+2} will be a non-linear T-box; Δm represents flips of one or both of the two middle bits of the internal S-box; and Δn represents flips of respectively one or two $\overline{R_{r+1}}$ bits.*

Property 4. *If Δv represents a flip of bits of $R_{r-1}\backslash\overline{R_{r-1}}$, there are exactly 2 propagated differences in round $r+2$. Both affected T-boxes are non-linear T-boxes, and each of their input differences will affect strictly more than two T-boxes in round $r+4$.*

Based on these properties, we have a tool to identify restricted bit flips, and to distinguish non-linear T-boxes. In Algorithm 2, this procedure 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 $\mathcal{S}_M(T_m^{r+2})$.

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 Δm in round $r+2$.
  $\delta n \leftarrow$ # affected T-boxes in round $r+4$ propagated by Δn 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:

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

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 $\Delta m_i : v_j^{r+2} \rightarrow v_j^{r+2} \oplus \Delta m_i$ the 3 generated differences. One can verify that, except for S-box S_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, the 4 output bits of an S-box affect a single input bit of 6 S-boxes in the next round, with two of them middle input bits (See Coppersmith [5]). Based on the previously mentioned study, the two differences representing bypass bits can be detected. Under the assumption of ordered bypass bits, we have already built this set to compare with ($\mathcal{S}_{\overline{R}}^{r+3}$). The other propagated input differences to the T-boxes in round $r+3$ represent

single bit flips. Algorithm 3 describes this procedure, which constructs the set $\mathcal{S}_S(T_i^{r+3})$ of differences representing single bit flips.

As mentioned, the described property does not hold for S_8: for the input $11b_4b_501$, with arbitrary b_4 and b_5, the rightmost output bit cannot be flipped by flipping the input bits b_4 and b_5. Thus, with a probability of 1/16, we are not able to find all single bit flips of round $r+3$. However, it will become clear in the next section that we do not need all information to successfully apply our cryptanalysis.

Algorithm 3. Finding single bit flips

```
1: for all Δv ∈ S_M(T^{r+2}_{π(j)})   j = 1...8 (for non-linear T-boxes) do
2:    Compute one round function
3:    for all Δw_i propagated difference to a non-linear T-box T_i^{r+3} do
4:       if Δw_i ∉ S_S(T_i^{r+3}) then
5:          Δw_i → S_S(T_i^{r+3})
6:       end if
7:    end for
8: end for
```

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 to find the value of its 6-bit input vector $(f_j^{r+3}|_{2\ldots7}(v_j^r) \oplus k_j^{r+3})$.

We define the set $\mathcal{P}(T_j^{r+3}) = \{(S_q, w_l) \mid 1 \leq q \leq 8, w_l \in GF(2)^6\}$ as the set of all possible pairs of S-boxes and input vectors. Our strategy is to remove all the invalid pairs from the set. We can do this by comparing 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}, with 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 define δ_i as the number of non-linear T-boxes that are affected in round $r+4$ when Δv_i is applied to the input of T_j^{r+3}. To verify a pair (S_q, w_l), we take part of a non-white-box DES implementation with S-box S_q and S-box input w_l, and simulate the behaviour of a flip of the i'th input bit to the S-box. Then, δ_i' is defined as the number of affected S-boxes in the next round of this simulation. Define Δw_i as the difference to the input of the internal S-box of the T-box to which Δv_i is applied $(\Delta w_i : w_l \rightarrow w_l \oplus f_j^{r+3}|_{2\ldots7}(\Delta v_i))$.

If (S_q, w_l) is a candidate solution, 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. δ_7' must be smaller or 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 $\{\delta'_3, \delta'_4\} \leq \{\delta_3, \delta_4\}$.
- Similarly $\{\delta'_1, \delta'_2, \delta'_5, \delta'_6\} \leq \{\delta_1, \delta_2, \delta_5, \delta_6\}$.

Any pair (S_q, w_l) that does not fulfil these conditions is removed from the set $\mathcal{P}(T_j^{r+3})$. At the end, if only pairs with one type S_q remain, then this S_q is the internal S-box of $T_j^{r+3}(\pi(q) = j)$. As soon as S-boxes are identified, we can also make use of S-box relations between 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 down the conditions $(\delta_2 = \delta'_2)$.

Because all 8 DES S-boxes are very different, and are highly non-linear, the filtering process will reduce most $\mathcal{P}(T_j^{r+3})$ sets 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.

3.4 Key Recovery

Given that we have found a sufficient number of inputs to S-boxes, we start an iterated recovery of key bits, initiated by guessing one single key bit, using the following two observations:

- The expansion operation E maps some of the input bits to 2 different S-boxes, prior to the key addition. From Sect. 3.3, we know the value of the input bits to these S-boxes, after the round key addition. Hence, if we know one of the corresponding two bits of the round key, we are able to compute the other key bit.
- The value of one single bit can be followed through several rounds. Consider an R_{r-1} bit. In round r and $r+2$, after the expansion and the round key addition, this is the (known) input to an S-box. In round $r+1$ it is XOR-ed with an output bit b of an S-box after the permutation P operation. Because P is known, the S-boxes in round $r+1$ are identified and their input is known, we can compute the value of b. Hence, if one bit of the round key bit in round r or $r+2$ 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, to which the two described methods can be applied.

Depending on the initial key bit guess, two complementary keys k_0 and k_1 can be computed. Because of the complementation property DES exhibits, both keys are a valid result. The complementation property of DES [12] 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 $G \circ DES_k \circ F = G' \circ DES_{k\oplus 1} \circ F'$, with $F' = \bigoplus_1 \circ F$ and $G' = G \circ \bigoplus_1$. Hence if k is the original DES key, and F, G the external encodings

used to shield the white-box DES implementation, then the complementary key $k \oplus 1$ is also a valid DES key with external encodings F', G'.

3.5 Recovery of the External Encodings

The main goal in cryptanalysis of white-box implementations is to find the embedded secret key. However, to break specific white-boxed implementations or decode ciphertext, recovery of the external encodings can be required.

These external encoding can be recovered as follows: for every input v_{EXTin} to the encoded implementation, we are able to find the inputs to the S-boxes. For Feistel ciphers, given the input to two consecutive rounds and the secret key, the plaintext can be computed easily. Hence we are able to compute the input to the naked DES, i.e., $v_{DESin} = F(v_{EXTin})$. Moreover, we can also compute the output of the naked DES, i.e., $v_{DESout} = DES_k(v_{DESin})$. This is the input to the external output encoding for which its output v_{EXTout} is the output of the white-box implementation. Hence for any given input to the white-box implementation, we can build different input-output pairs of the external encodings. This way, with a sufficient number of chosen inputs, the external encodings can be computed. Here we assume that these encodings are not too complex, that is, rather affine or simple non-linear mapping.

Chow *et al.* [4] proposed a specific class of external encodings, which are block encoded affine mixing bijections. Suppose these block encodings are nibble encodings. Then, for each of the 24 nibble encodings, we run over all its possible inputs (2^4), and compute the value of the 96-bit output v_{DESin}. With the knowledge of all these mappings, we are able to recover the external input encoding. The external output encoding can be recovered similarly.

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 are needed to be computed to check the difference propagations. This is less than our complexity study in Appendix B indicates, due to some extra optimisations we have applied (e.g., introducing requirements regarding round $r+1$ in Property 1 substantially improves the efficiency of the algorithm). Moreover, our tests indicate that computations with 8 consecutive obfuscated round functions is sufficient for the attack to succeed. There is no restriction on which window of 8 round functions to chose.

In the conducted tests on several white-box DES implementations, our cryptanalysis algorithm extracted the DES key in all tests in under a second. On average the cryptanalysis requires 0.64 seconds.

5 Conclusion

We have described how to extract the embedded secret key of both the naked as encoded white-box DES implementations of Chow *et al.* [4]. This cryptanalysis

also applies to the improved implementation as presented by Link and Neumann [10], because the outputs of the T-boxes are not used, only their inputs. The attack is a differential cryptanalysis on the obfuscated rounds, and is independent of the definition of the external encodings, in contrast to the attack of Goubin *et al.* [6].

The success of this cryptanalysis originates from properties which are specific to the 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 used to extract the key. The analysis, while specific to DES, nevertheless points the way to techniques to analyse other ciphers.

Acknowledgements. This work has been developed jointly with Philips Research Laboratories and is partly funded by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen); and the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, the Research Foundation - Flanders (FWO-Vlaanderen). We would also like to thank Dries Schellekens and the anonymous reviewers for their constructive comments.

References

1. Biham, E., Shamir, A.: Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer (extended abstract). In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 156–171. Springer, Heidelberg (1992)
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. Chow, S., Eisen, P.A., 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)
4. Chow, S., Eise, P.A., 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. Coppersmith, D.: The Data Encryption Standard (DES) and its strength against attacks. IBM J. Res. Dev. 38(3), 243–250 (1994)
6. Goubin, L., Masereel, J.-M., Quisquater, M.: Cryptanalysis of white box DES implementations. Cryptology ePrint Archive, Report 2007/035 (2007), `http://eprint.iacr.org/`
7. Jacob, M., Boneh, D., Felten, E.W.: Attacking an obfuscated cipher by injecting faults. In: Digital Rights Management Workshop, pp. 16–31 (2002)
8. Kerins, T., Kursawe, K.: A cautionary note on weak implementations of block ciphers. In: WISSec 2006. 1st Benelux Workshop on Information and System Security, Antwerp, BE, p. 12 (2006)
9. Kim, J., Hong, S., Preneel, B.: Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In: Fast Software Encryption (2007)
10. Link, H.E., Neumann, W.D.: Clarifying obfuscation: Improving the security of white-box DES. In: ITCC 2005. Proceedings of the International Conference on Information Technology: Coding and Computing, vol. I, pp. 679–684. IEEE Computer Society Press, Washington, DC, USA (2005)

11. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: The case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)
12. Pfleeger, C.P.: Security in Computing. Prentice-Hall, Englewood Cliffs, New Jersey (1989)
13. Data Encryption Standard. `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`

A Appendix: Proofs

Property 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 flip of a single bit of R_{r-1}, then in round $r+1$ it represents a flip of single bit of L_r, 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. □

Property 2. *If Δv represents flips of bits of L_{r-1} or Y_r, then in almost all cases more than 2 T-boxes are affected in round $r+2$. The exceptions (false positives) can be detected by repeating this process up to α times with different fixed inputs to the other T-boxes T_l^r.*

Proof. In round $r+1$, besides bypass bits, these differences represent flips to the inputs of S-boxes. Therefore, the number of flips to the inputs of round $r+2$ explodes, and strictly more than 2 T-boxes will be affected.

There are a few exceptions in which not more than 2 T-boxes are affected (*false positives*). Observe an affected S-box in round $r+1$. (There will always be at least one affected S-box). The input to this S-box changes in at least one and at most 3 bits (one for Y_r and two for L_{r-1} bit flips). 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. Hence the number of affected T-boxes in round $r+2$ will very likely change if we set other inputs to T_l^r, witch $l \neq j$. With a very high probability, 2 extra checks are sufficient to detect these false positives, if we change all the inputs to the other T-boxes ($\alpha = 2$). □

Property 3. *If Δv represents a flip of single bits of $\overline{R_{r-1}}$, there are exactly 2 propagated differences in round $r+2$: Δm, Δn. One (say Δm, input difference to T-box T_m^{r+2}) will affect strictly more than 2 T-boxes in round $r+4$, the other difference will affect at most 2 T-boxes in round $r+4$. Moreover, T_m^{r+2} will be a non-linear T-box; Δm represents flips of one or both of the two middle bits of the internal S-box; and Δn represents flips of respectively one or two $\overline{R_{r+1}}$ bits.*

Proof. Let $\Delta v \in \mathcal{S}_{\overline{R}}(T_j^r)$ represent a flip of single (or double) bits of $\overline{R_{r-1}}$. Then, in round $r+2$, this will propagate to a flip of one (or both) of the two middle input bits of an S-box S_m in T-box T_m^{r+2}. Hence T_m^{r+2} is a non-linear

T-box. 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 an entropy of 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]:

$$\Delta_{in} = 0wxyz0 \Rightarrow |\Delta_{out}| \geq 2 \quad (1)$$

$$|\Delta_{in}| = 1 \Rightarrow |\Delta_{out}| \geq 2\,, \quad (2)$$

with Δ_{in} the input difference to an S-box, Δ_{out} its resulting output difference, and $wxyz \in GF(2)^4 \backslash \{0\}$. 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 affects no more than two T-boxes in round $r+4$ (see Property 1). □

Property 4. *If Δv represents a flip of bits of $R_{r-1} \backslash \overline{R_{r-1}}$, there are exactly 2 propagated differences in round $r+2$. Both affected T-boxes are non-linear T-boxes, and each of their input differences will affect strictly more than two T-boxes in round $r+4$.*

Proof. If $\Delta v \in \mathcal{S}_R(T_j^r)$ represents a flip of bits of $R_{r-1} \backslash \overline{R_{r-1}}$, then for 2 S-boxes in round $r+2$, exactly one input bit will be affected, and thus exactly 2 non-linear T-boxes in round $r+2$ are affected.

Because of S-box design property (2), each of these differences will represent a flip of at least two Y_{r+2} bits. As a consequence of the DES permutation P diffusion property, both these differences in round $r+2$ will affect strictly more than two T-boxes in round $r+4$. □

B 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 first step described, to retrieve flips of bits of R_{r-1}, has the largest complexity. Because of the lack of any prior information on internal flips, all differences have to be computed through several rounds in order to learn this bit flip information.

In Algorithm 1, for all 12 T-boxes, and all $2^8 - 1$ possible differences, 2 rounds need to be computed to observe the difference propagation. This corresponds to a total of $12 \cdot (2^8 - 1) \cdot 2 = 6120$ round function computations. For each positive result, we perform at most 2 double checks as described in Property 2. Algorithm 2 requires 6 round computations for each difference of $\mathcal{S}_R$ (2 for Δv, 2 for Δl and 2 for Δm). Hence, 240 round functions computations are performed.

Consequently, we can retrieve all flips of bits of $\overline{R_{r-1}}$ for one round in less than 2^{13} round computations in total. As described in Property 2, from $\Delta v \in \mathcal{S}_{\overline{R}}^r$, we

can efficiently compute $\Delta n \in \mathcal{S}_{\overline{R}}^{r+2}$. Because of the one-to-one relation between Δv and Δn, this is sufficient to find all the single $\mathcal{S}_{\overline{R}}^{r+2}$ bit flips. Thus, when for two consecutive rounds, $\mathcal{S}_{\overline{R}}$ is found, we can compute this set for all subsequent rounds using Property 3 only. Hence, with about 2^{14} round computations, we can compute all flips of single bits of $\overline{R_{r-1}}$ for all rounds.

The complexity of the other steps of the cryptanalysis is negligible. In Algorithm 3, for each $\Delta m \in \mathcal{S}_M^r$, one round function needs to be computed. Hence, for each round, at most 24 round computations are needed (for 16 single bit flips and at most 8 double bit flips). To compute the exact inputs to the S-boxes, a filtering process needs to be applied to each non-linear T-box. In the worst case, we need to compute the difference propagation for all 7 input differences. Thus at most 7 round computations for each of the 8 non-linear T-boxes. 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, which is the equivalent effort of computing one white-box DES round function (which consists of $552 \sim 2^9$ lookup table computations). The total complexity to compute the inputs to all S-boxes of one round is thus $8 \cdot (7 + 1) = 2^6$.

The space complexity is negligible too. 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 the simulation does not require any information on the implementation or the key.