Information Processing Letters 114 (2014) 252–255

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

On computational complexity of impossible differential cryptanalysis

Mohsen Shakiba [a,*], Mohammad Dakhilalian [a], Hamid Mala [b]

[a] Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran
[b] Department of Information Technology Engineering, University of Isfahan, Isfahan, Iran

ARTICLE INFO

Article history:
Received 18 July 2013
Received in revised form 30 November 2013
Accepted 17 December 2013
Available online 18 December 2013
Communicated by V. Rijmen

Keywords:
Cryptography
Block cipher
Impossible differential cryptanalysis
Computational complexity

ABSTRACT

Impossible differential cryptanalysis is one of the conventional methods in the field of cryptanalysis of block ciphers. In this paper, a general model of an impossible differential attack is introduced. Then, according to this model, the concept of an *ideal* impossible differential attack is defined and it is proven that the time complexity of an *ideal* attack only depends on the number of *involved* round key bits in the attack.

1. Introduction

Impossible differential cryptanalysis, an extension of the differential attack [1], was first introduced by Knudsen and Biham to analyze DEAL [2] and Skipjack [3], respectively. This cryptanalysis method have achieved significant results on many well-known block ciphers including the attacks on AES-128 [4–6], Camellia [7–10], ARIA [7,11] and MISTY-1 [12–14].

As it is known, the time complexity of an impossible differential attack is determined based on the attack procedure. According to the published instances of this cryptanalysis method, it seems that the time complexity can be reduced by a variants of different techniques, including early abort technique, hash tables and key scheduling considerations. This paper discusses the minimal computational complexity of an impossible differential attack and introduce an expression for the minimum possible time complexity which only depends on the number of round key bits involved in the attack.

The block cipher E which is considered in this paper is an iterative symmetric block cipher that consists of several sequential rounds. Each round of E contains one Add-Round-Key operation which is assumed to be a modulo 2 addition (bitwise XOR), and a *PS*-box including some nonlinear substitutions and linear permutations. Thus a round function maps an input x to $PS(x \oplus k)$ under a round key k. This representation covers all types of block cipher structures with bit-wise XOR key addition, whether they are Feistel ciphers such as Camellia [15], or Substitution–Permutation Networks (SPN) such as AES [16], or even block ciphers of other structures such as FOX [17]. The block cipher E encrypts a plaintext under a secret key K to obtain its corresponding ciphertext. Each round key is determined as a (direct or recursive) function of the secret key.

The paper is organized as follows. In Section 2 the general form of an impossible differential attack is introduced. Then, the concept of an *ideal attack* and a tight approximation of the attack's time complexity is discussed in Section 3. Finally, the paper is concluded with two examples on AES and Camellia in Section 4.

* Corresponding author.
E-mail addresses: m.shakiba@ec.iut.ac.ir (M. Shakiba), mdalian@cc.iut.ac.ir (M. Dakhilalian), h.mala@eng.ui.ac.ir (H. Mala).

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2013.12.008

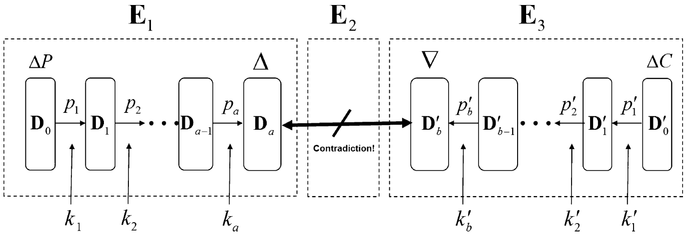

Fig. 1. General structure of an impossible differential attack.

2. A general model for impossible differential attack

As it is indicated in Fig. 1, block cipher E is divided into three sub-ciphers E_1, E_2 and E_3, in an impossible differential attack. There is an impossible differential for the sub-cipher E_2 which is defined by two sets of differences in its input and output, called Δ and ∇. Each difference $d \in \Delta$ is in contradiction with every difference in ∇. In other words, for any arbitrary secret key K, it is impossible to have a pair (x_1, x_2) and its corresponding pair (y_1, y_2) $(y_i = E_2(x_i, K))$ such that $x_1 \oplus x_2 = d \in \Delta$ and $y_1 \oplus y_2 = d' \in \nabla$.

In sub-cipher E_1 we define a sequence of sets of differences including an input difference D_0 and differences D_1 to D_a in the outputs of rounds of E_1 (Note that $D_a \subseteq \Delta$). For a pair (x_1, x_2) which $x_1 \oplus x_2 \in D_{i-1}$, after one-round encryption under an arbitrary round-key value k_i, its corresponding output pair (y_1, y_2) will have a difference $y_1 \oplus y_2 \in D_i$ with probability p_i. Thus, probabilities p_i, $1 \leqslant i \leqslant a$, are the difference transition probabilities in the encryption path. In the same way, for sub-cipher E_3, we define a sequence of sets of differences D'_0 to D'_b $(D'_b \subseteq \nabla)$ and the difference transition probabilities p'_j, $1 \leqslant j \leqslant b$, in the decryption path.

By concatenating the whole round keys in sub-ciphers E_1 and E_3 we define sub-cipher keys $K_{E_1} = k_1|k_2|\cdots|k_a$ and $K_{E_3} = k'_1|k'_2|\cdots|k'_b$, respectively. However, in the procedure of an impossible differential attack, we need only those bits of K_{E_1} and K_{E_3} which are *required* to check whether the intermediate differences meet the expected differences $D_0, \ldots, D_a, D'_0, \ldots, D'_b$ or not. Also, according to the key schedule of E, sometimes we are able to determine the values of some *required key bits* of a round key k_i as a function of the *required key bits* that have already been guessed. Such bits are *redundant key bits*. Those bits of k_i which are *required* and also are not *redundant* are called *involved key bits* and are indicated by k_i^l. *Involved key bits* of sub-ciphers E_1 and E_3 are indicated by $K^l_{E_1} = k^l_1|k^l_2|\cdots|k^l_a$ and $K^l_{E_3} = k'^l_1|k'^l_2|\cdots|k'^l_b$. The goal of the attack is to discover the correct value of the whole of *involved key bits* $K^l = K^l_{E_1}|K^l_{E_3}$.

2.1. General attack procedure

We call a pair of plaintexts (P_0, P_1), $P_0 \oplus P_1 \in D_0$, a *proper pair* if their corresponding ciphertext difference $C_0 \oplus C_1$ meets D'_0. Assume that K^l consists of L bits $(L = |K^l|)$. If we initialize a list of all 2^L possible values of K^l, then for each proper pair, those values of K^l which satisfy all of the $a + b$ intermediate differences are certainly incorrect and must be eliminated from the list. If there are enough proper pairs, which is assumed to be 2^N pairs, then this process lasts until only 2^{ε} candidates remain in the list. The correct value of K^l is certainly one of the remaining candidates. However, bits of K^l are guessed in a pre-defined order. Due to the order of the bits of K^l, the attack procedure can be performed step by step, where each step corresponds to some bits of one k^l_i (k'^l_j) which value can be guessed independent of the other bits of k^l_i (k'^l_j). Thus, K^l is divided among the steps and for t-th step of the attack, $1 \leqslant t \leqslant T$, the corresponding bits of K^l are indicated by λ_t. Also, α_t is the difference transition probability of step t. As it will be explained in the following, each step of the attack can be done using encryptions or memory accesses based on pre-computed tables.

Encryption based procedure In the first step of the attack we perform $Z_1 = 2^{|\lambda_1|}$ (partial) encryptions to obtain $S_1 = \alpha_1 \times 2^{|\lambda_1|}$ values for λ_1. Suppose in step $t-1$, $t \geqslant 2$, we obtain S_{t-1} values of $\lambda_1|\lambda_2|\cdots|\lambda_{t-1}$ which satisfy all of the corresponding intermediate differences until the current step. Then, in t-th step of the attack we perform $Z_t = S_{t-1} \times 2^{|\lambda_t|}$ encryptions to obtain $S_t = S_{t-1} \times (\alpha_t \times 2^{|\lambda_t|})$ values of $\lambda_1|\lambda_2|\cdots|\lambda_{t-1}|\lambda_t$ which also satisfy the intermediate difference associated with the t-th step of the attack.

Suppose, $\prod_{t=1}^{T} \alpha_t = \prod_{i=1}^{a} p_i \times \prod_{j=1}^{b} p'_j$ is denoted by 2^Q. Also, as it is expected we have $\sum_{t=1}^{T} |\lambda_t| = \sum_{i=1}^{a} |k^l_i| + \sum_{j=1}^{b} |k'^l_j| = L$. Thus, in the last step of the attack, for each proper pair, we perform $Z_T = S_{T-1} \times 2^{|\lambda_T|} = 2^{|\lambda_T|} \times \prod_{t=1}^{T-1} \alpha_t \times 2^{|\lambda_t|} = (1/\alpha_T) \times 2^{Q+L}$ encryptions to obtain $S_T = Z_T \times \alpha_T = 2^{Q+L}$ values of K^l which must be eliminated from the list. The time complexity is dominated by the step m with the largest Z_m.

Pre-computation based procedure Each step of the attack can be done by pre-computation. For this purpose, in an offline stage, corresponding to step t of the attack, by decryption (encryption) of possible pairs in the output (input) of *PS*-box (according to the output difference), those pairs which differences satisfy the input (output) difference are obtained and stored in a row of a hash table indexed by the input differences. So, as it is expected, in each row of such a hash table, about $\alpha_t \times 2^{|\lambda_t|}$ pairs are stored. As a result, in the online stage of the attack, step t is done by S_t memory accesses (for each proper pair) to obtain S_t values of $\lambda_1|\lambda_2|\cdots|\lambda_{t-1}|\lambda_t$. Thus, the time complexity of the last step is reduced to $S_T = 2^{Q+L}$ memory accesses. It must be noted that there is no guarantee for