Differential Fault Analysis: Identifying the Structure of Unknown Ciphers Sealed in Tamper-Proof Devices

Eli Biham
Adi Shamir
Computer Science Dept.

Applied Math Dept.
Technion
The Weizmann Institute
Israel
Israel

November 10, 1996
(DRAFT)

In recent announcements we described how to break DES and to find keys of unknown ciphers on tamper-proof devices, when the attacker can induce faults during encryption. We used a fault model similar to the model suggested by Boneh, Demillo and Lipton. In this announcement we study this topic further: we show that in most interesting cases we can identify the structure of an unknown cipher sealed in the tamper-proof device (e.g., SkipJack), including the identification of its round functions, S boxes, and subkeys. If the attacker can only encrypt with the tamper-proof device with a fixed key (as is applied on PIN verifying devices), still the attacker can identify the operation of the cipher with this particular key, which lets him to encrypt and decrypt under this key.

In this announcement we describe the application to Feistel (DES-like) ciphers. In the final paper we will also study the more general case.

We should first identify which ciphertext bits come from the right half and which from the left half of the last round, i.e., which bits affect which in the last round. We encrypt a plaintext several times, and collect pairs of ciphertexts consisting of the real ciphertext of the plaintext and a faulty ciphertext resulting from some fault during encryption. We identify the faults which occur in the last round (or two) by counting the bits differing between the real ciphertext and the faulty ciphertext. We use the pairs in which the number of differing bits in the ciphertexts is smaller than a threshold (typically about a quarter of the blocksize), in which case, it is almost certain that the fault occurred in the last round, or in the preceding one. Each fault in the last round differs in one bit in the right half and several bits in the left half. Since there are only 32 such possible faults in the last round, we receive at most 32 such faulty ciphertexts. Each bit of the right half differs only in one of these ciphertexts, and each bit of the left half typically differs in several of the ciphertexts. We can thus identify the bits of the right half as those which differ in the least number of pairs.

Once we identified the bits of the right half and the bits of the left half, we can observe which bits of the left half are affected by each bit of the right half. In case that the cipher is similar to DES in that the F function is composed from S boxes, each takes a few of the bits of the right half, and affects a few of the bits of the left half, we can easily identify the number of S boxes, and the input and output bits of each S box: we just choose all the pairs which differ by only one bit of the right half, and keep for each such bit the set of all the bits of the left half which differ in those pairs. This information suffices to find the number of S boxes, their sizes, and to identify the input and output bits of each S box.

Then, it is not difficult to reconstruct the content of the S boxes, with the specific used key mixed into the input of the S boxes and some unknown value mixed into their output (i.e., we can identify the table $T(x)$, where $T(x)=S(x XOR k) XOR u$, where k is the subkey, and u is derived from the right half of the preceding round). This reconstruction is very effective. For example, if the unknown cipher is DES, we miss information on only $6+4=10$ bits out of the $64*4=256$ unknown bits of each S box, and if the unknown cipher is LOKI, we miss information on only $12+8=20$ bits out of the $2^{12}*8=32768$ bits of each S box of LOKI. These missing bits do not reduce the success of the attack since we actually find all the information we need for peeling off the last round; the extra constants are counted naturally as parts of the subkeys when analyzing the preceding

rounds. This way we can fully analyze the whole cipher, and receive its full description with the specific key mixed into the S boxes.

Till now we identified the S boxes up to some unknown XORs, some of which are subkeys. We can further identify the S boxes and the subkeys by analyzing encryptions under several keys and comparing the differences between the retrieved tables T. In DES and LOKI the key scheduling algorithm and the S boxes can be easily identified by such comparisons.

If some ciphers, the F function is not a composition of several S boxes as in DES and LOKI, but may compute more complex operations. In this case we can model the F function by a more general structure. We can even identify the F function when it is modeled by the very general form of XORing the right half and the subkey (possibly after an expansion), and perform any function on the output. In this case we can view the F function as applying one huge S box. The number of input and output bits of this huge S box is only half the block size, and thus even though the identification of this whole S box requires much work, it is still practical. In comparison, searching exhaustively over all the possible unknown ciphers and over all their possible keys to identify the structure of the unknown cipher would require almost infinite number of operations even if we consider only very simple ciphers with only a few rounds.

We have implemented major parts of this attack on a personal computers, using DES as the unknown cipher. Our implementation required only about 500 faulty ciphertexts to identify the bits of the right half, and up to 5000 faulty ciphertexts to identify the S boxes and their input and output bits, without reconstructing the actual entries of the S boxes. This reconstruction of the S boxes should require about 10000 faulty ciphertexts.

Note that the complexity of this S box reconstruction crucially depends on the size of the S boxes: in DES there are 64 entries in each S box, and thus about 64 faults in the input of an S box in different ciphertexts suffice to reconstruct the S box (up to the 10 missing bits). In LOKI there are 4096 entries in each S box, and thus the number of required faulty ciphertexts is much larger. If we view the F function as one large S box, the number of entries of this huge S box is 2^{32}, and thus the number of required faulty ciphertexts in this case is huge, but still practical.

In the final paper we will elaborate on the above attacks, their properties, and their implementations, and will discuss the consequences to more general classes of ciphers.