Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials

Eli Biham* Alex Biryukov** Adi Shamir***

Abstract. In this paper we present a new cryptanalytic technique, based on *impossible differentials*, and use it to show that Skipjack reduced from 32 to 31 rounds can be broken by an attack which is faster than exhaustive search.

Key words. Skipjack, Cryptanalysis, Differential cryptanalysis, Impossible differentials.

1 Introduction

Differential cryptanalysis [6] traditionally considers characteristics or differentials with relatively high probabilities and uses them to distinguish the correct unknown keys from the wrong keys. When a correct key is used to decrypt the last few rounds of many pairs of ciphertexts, it is expected that the difference predicted by the differential appears frequently, while when a wrong key is used the difference occurs less frequently.

In this paper we describe a new variant of differential cryptanalysis in which a differential predicts that particular differences should not occur (i.e., that their probability is exactly zero), and thus the correct key can never decrypt a pair of ciphertexts to that difference. Therefore, if a pair is decrypted to this difference under some trial key, then certainly this trial key is not the correct key. This is a *sieving attack* which finds the correct keys by eliminating all the other keys which lead to contradictions.

We call the differentials with probability zero *Impossible differentials*, and this method of cryptanalysis *Cryptanalysis with impossible differentials*.

* Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel, biham@cs.technion.ac.il, http://www.cs.technion.ac.il/~biham/.

** Applied Mathematics Department, Technion – Israel Institute of Technology, Haifa 32000, Israel, albi@cs.technion.ac.il.

*** Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel, shamir@wisdom.weizmann.ac.il.

We should emphasize that the idea of using impossible events in cryptanalysis is not new. It is well known [7] that the British cryptanalysis of the German Enigma in world war II used several such ideas (for example, a plaintext letter could not be encrypted to itself, and thus an incorrectly guesses plaintext could be easily discarded). The first application of impossible events in differential cryptanalysis was mentioned in [6], where zero entries in the difference distribution tables were used to discard wrong pairs before the counting phase. A more recent cryptanalytic attack based on impossible events was described by Biham in 1995 in the cryptanalysis of Ladder-DES, a 4-round Feistel cipher using DES as the F function. This cryptanalysis was published in [3], and was based on the fact that collisions cannot be generated by a permutation. A similar technique was latter used by Knudsen in his description of DEAL [8], a six-round Feistel cipher using DES as the F function. Although the idea of using impossible events of this type was natural in the context of Feistel ciphers with only a few rounds and with permutations as the round function, there was no general methodology for combining impossible events with differential cryptanalytic techniques, and for generating impossible differentials with a large number of rounds.

In this paper we show that cryptanalysis with impossible differentials is very powerful against many ciphers with various structures. We describe an impossible differential of Skipjack which ensures that for all keys there are no pairs of inputs with particular differences with the property that after 24 rounds of encryption the outputs have some other particular differences. This differential can be used to attack Skipjack reduced to 31 rounds (i.e., Skipjack from which only the first or the last round is removed), slightly faster than exhaustive search, and to distinguish whether a black box applies a 24-round variant of Skipjack, or a random permutation. In a related paper [5] we describe the application of this type of cryptanalysis to IDEA [10] and to Khufu [12], which improves the best known attacks on these schemes.

For conventional cryptanalysis of Skipjack with smaller numbers of rounds we refer the reader to [4] and to [9].

The paper is organized as follows: The description of Skipjack is given in Section 2. The 24-round impossible differential of Skipjack is described in Section 3. In Section 4 we describe a simple variant of our attack against Skipjack reduced to 25 and to 26 rounds, and in Section 5 we describe our main attack applied against Skipjack reduced to 31 rounds. Finally, in Section 6 we discuss why the attack is not directly applicable to the full 32-round Skipjack, and summarize the paper.

Rule A	RuleB
$w_1^{k+1} = G^k(w_1^k) \oplus w_4^k \oplus counter^k$	$w_1^{k+1} = w_4^k$
$w_2^{k+1} = G^k(w_1^k)$	$w_2^{k+1} = G^k(w_1^k)$
$w_3^{k+1} = w_2^k$	$w_3^{k+1} = w_1^k \oplus w_2^k \oplus counter^k$
$w_4^{k+1} = w_3^k$	$w_4^{k+1} = w_3^k$

Fig. 1. Rule A and Rule B.

2 Description of Skipjack

Skipjack is an iterated blockcipher with 32 rounds of two types, called Rule A and Rule B. Each round is described in the form of a linear feedback shift register with additional non linear keyed G permutation. Rule B is basically the inverse of Rule A with minor positioning differences. Skipjack applies eight rounds of Rule A, followed by eight rounds of Rule B, followed by another eight rounds of Rule A, followed by another eight rounds of Rule B. The original definitions of Rule A and Rule B are given in Figure 1, where *counter* is the round number (in the range 1 to 32), and where G is a four-round Feistel permutation whose F function is defined as an 8x8-bit S box, and each round of G is keyed by eight bits of the key. The key scheduling of Skipjack takes a 10-byte key, and uses four of them at a time to key each G permutation. The first four bytes are used to key the first G permutation, and each additional G permutation is keyed by the next four bytes cyclically, with a cycle of five rounds.

The description becomes simpler if we unroll the rounds, and keep the four elements in the shift register stationary. Figure 2 describes this representation of Skipjack (only the first 16 rounds out of 32 are listed; the next 16 rounds are identical except for the counter values). The unusual structure after round 8 (and after round 24) is the result of simplifying the two consecutive XOR operations at the boundary between Rule A and Rule B rounds.

3 A 24-Round Differential with Probability Zero

We concentrate on the 24 rounds of Skipjack starting from round 5 and ending at round 28 (i.e., without the first four rounds and the last four rounds). For the sake of clarity, we use the original round numbers of the full Skipjack, i.e., from 5 to 28, rather than from 1 to 24. Given any pair with difference only in the second word of the input of round 5, i.e., with a difference of the form $(0, a, 0, 0)$, the difference after round 28 cannot be of the form $(b, 0, 0, 0)$, for any non-zero a and b. (We are aware of several shorter differentials with probability zero; this one is the longest known to us).

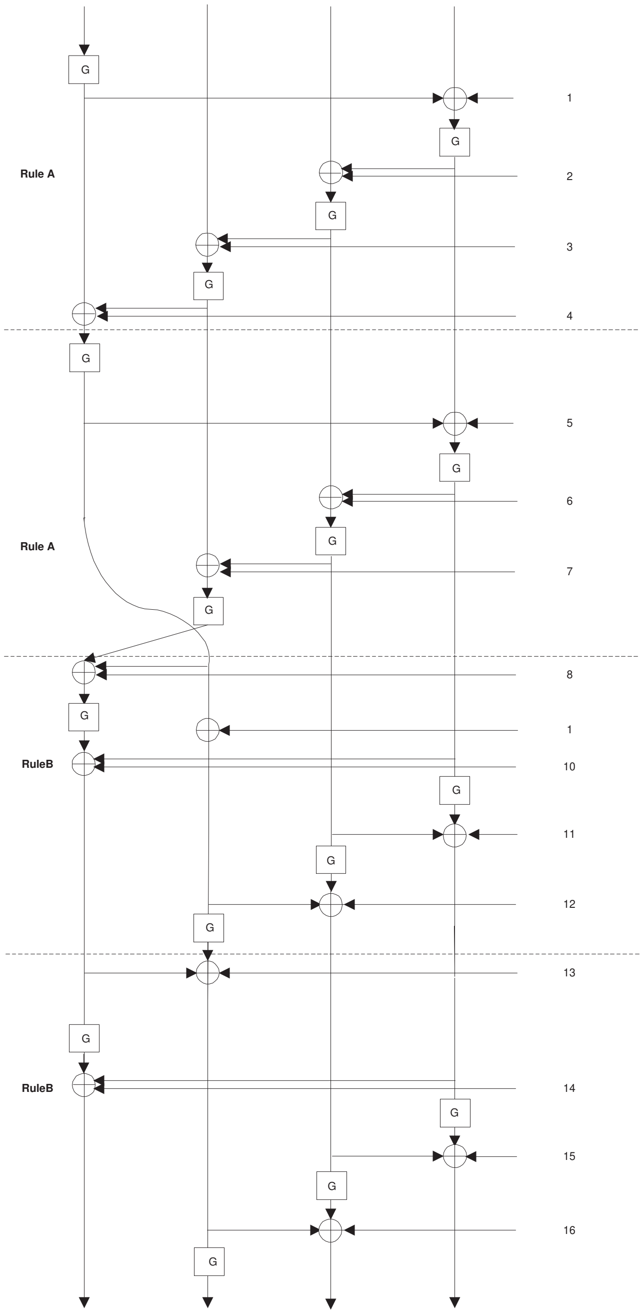

Fig. 2. Skipjack.

The reason that this differential has probability zero can be explained as a *miss in the middle* combination of two 12-round differentials with probability 1: As Wagner observed in [16], the second input word of round 5 does not affect the fourth word after round 16, and given an input difference $(0, a, 0, 0)$ the difference after 12 rounds is of the form $(c, d, e, 0)$ for some non-zero c, d, and e. On the other hand, we can predict the data after round 16 from the output difference of round 28, i.e., to consider the differentials in the backward direction. Similarly to the 12-round differential with probability 1, there is a backward 12-round differential with probability 1. It has the difference $(b, 0, 0, 0)$ after round 28, and it predicts that the data after round 16 must be of the form $(f, g, 0, h)$ for some non-zero f, g, and h. Combining these two differentials, we conclude that any pair with difference $(0, a, 0, 0)$ after round 4 and difference $(b, 0, 0, 0)$ after round 28 must have differences of the form $(c, d, e, 0) = (f, g, 0, h)$ after round 16 for some non-zero c, d, e, f, g, and h. As e and h are non-zero, we get a contradiction, and thus there cannot be pairs with such differences after rounds 4 and 28.

One application of this differential may be to distinguish whether an encryption black box is a 24-round Skipjack (from round 5 to round 28), or a random permutation. Identification requires only to feed the black box with $2^{48}\alpha$ pairs (for some α) with differences of the form $(0, a, 0, 0)$, and to verify whether the output differences are of the form $(b, 0, 0, 0)$. If for some pair the output difference is of the form $(b, 0, 0, 0)$, the black box certainly does not apply this variant of Skipjack. On the other hand, if the black box implements a random permutation, there is only a probability of $e^{-\alpha}$ that none of the $2^{48}\alpha$ pairs has a difference $(b, 0, 0, 0)$. For example, given 2^{52} pairs the probability of the black box to be incorrectly identified as this variant of Skipjack is only $e^{-16} \approx 10^{-7}$. These pairs can be packed efficiently using structures of 2^{16} plaintexts which form 2^{31} pairs. In these structures all the plaintexts are equal except for the second word which ranges over all the possible 2^{16} values. Using these structures, the same distinguishing results can be reached using only $2^{33}\alpha$ encryptions.

4 Attack on Skipjack Reduced to 25-26 Rounds

In this section we describe the simplest cryptanalysis of Skipjack variants, with only one or two additional rounds (on top of the 24-round impossible differential itself). An attack on a 25-round variant of Skipjack from round 5 to round 29 is as follows. Choose structures of 2^{16} plaintexts which differ only at their second word, having all the possible values in it. Such structures propose about 2^{31} pairs of plaintexts. Given 2^{22} such structures (2^{38} plaintexts), collect all those pairs which differ only at the first two words of the ciphertexts; by the structure of Skipjack, only these pairs may result from pairs with a difference $(b, 0, 0, 0)$ after round 28. On average only half of the structures propose such pairs, and thus only about 2^{21} pairs remain. Denote the ciphertexts of such a pair by (C_1, C_2, C_3, C_4)

and (C_1^*, C_2^*, C_3, C_4). The pair may have a difference of the form $(b, 0, 0, 0)$ before the last round only if the decrypted values of C_1 and C_1^* by the G permutation in the last round have difference $C_2' = C_2 \oplus C_2^*$. As we know that such a difference is impossible, every key that proposes such a difference is a wrong key. For each pair we try all the 2^{32} possible values of the subkey of the last round, and verify whether the decrypted values by the last G permutation have the difference C_2' (this process can be done efficiently in about 2^{16} steps). It is expected that about 2^{16} values propose this difference, and thus we are guaranteed that these 2^{16} values are not the correct subkey of the last round. After analyzing the 2^{21} pairs, there remain only about $2^{32} \cdot (1-2^{-16})^{2^{21}} = 2^{32} \cdot e^{-32} \approx 2^{-14}$ wrong values of the subkey of the last round. It is thus expected that only one value remains, and this value must be the correct subkey. The time complexity of recovering this last 32-bit subkey is about $2^{17} \cdot 2^{21} = 2^{38}$ G permutation computations. Since each encryption consists of about 2^5 applications of G, this time complexity is equivalent to about 2^{33} encryptions. A straightforward implementation of the attack requires an array of 2^{32} bits to keep the information of the already identified wrong keys. A more efficient implementation requires only about 2^{32} G computations on average, which is about 2^{27} encryptions, and using 2^{16} bits of memory.

Essentially the same attack works against a 26-round variant from round 4 to round 29. In this variant, the same subkey is used in the first and last rounds. The attack is as follows: Choose 2^6 structures of 2^{32} plaintexts which differ only in the first two words and get all the 2^{32} values of these two words. Find the pairs which differ only in the first two words of the ciphertexts. It is expected that about $2^6 \cdot 2^{63}/2^{32} = 2^{37}$ pairs remain. Each of these pairs propose one wrong subkey value on average, and thus with a high probability after analysis of all the pairs only the correct first/last subkey remains. The time complexity of this attack when done efficiently is 2^{48}, using an array of 2^{16} bits. The rest of the key bits can be found by exhaustive search of 2^{48} keys.

5 Cryptanalysis of Skipjack Reduced to 31 Rounds

For the cryptanalysis of Skipjack reduced to 31 rounds, we use again the 24-round impossible differential. We first analyze the variant consisting of the first 31 rounds of Skipjack, and then the variant consisting of the last 31 rounds of Skipjack.

Before we describe the full details of the attack, we wish to emphasize several delicate points. We observe that the full 80-bit key is used in the first four rounds (before the differential), and is also used in the last three rounds (after the differential). Therefore, the key-elimination process should discard 80-bit candidate keys. Assuming that the verification of each of the 2^{80} keys costs at least one G computation, and as one G computation is about 31 times faster than one encryption, we end up with an attack whose time complexity is at least $2^{80}/31 \approx 2^{75}$ encryptions. This lower bound is only marginally smaller than exhaustive search, and therefore the attack cannot spend more than a few G operations verifying each key, and cannot try each key more than a few times.

We next observe that if the impossible differential holds in some pair, then the third word of the plaintexts and the third and fourth words of the ciphertexts have zero differences, and the other words have non-zero differences. Given a pair with such differences, and assuming that the differential holds, we get three 16-bit restrictions in rounds 1, 4, and 29. Therefore, we expect that a fraction of 2^{-48} of the keys, i.e., about 2^{32} keys, encrypt the plaintext pair to the input difference of the differential after round 4, and decrypt the ciphertext pair to the output difference of the differential before round 29. Once verified, these keys are discarded. These 2^{32} keys must be discarded with complexity no higher than 2^{32} as we mentioned earlier. Thus, we cannot try all the 2^{80} keys for each pair, but rather, we devise an efficient algorithm to compute the 2^{32} keys.

The general structure of the attack is thus expected to be as follows: we generate a large structure of chosen plaintexts and select the pairs satisfying the required differences. We analyze these pairs, and each of them discards about 2^{32} keys. After the analysis of 2^{48} pairs, about 2^{80} (not necessarily distinct) keys are discarded. We expect that due to collisions, about $1/e$ of the keys remain undiscarded. The analysis of additional pairs decreases the number of undiscarded keys, until after about $2^{48} \ln 2^{80} \approx 2^{48} \cdot 2^{6}$ pairs only the correct key remains. However, the complexity of such an attack is higher than the complexity of exhaustive search.

Therefore, we analyze only 2^{49} pairs, leaving about $2^{80}/e^2 \approx 2^{77}$ keys undiscarded, and then try the remaining keys exhaustively. We emphasize that the analysis discards keys which cause partial encryption and decryption of a valid pair to match the form of the impossible differential. We thus assume in the attack that the differences proposed by the impossible differential do hold, and discard all keys which confirm this false assumption.

We are now ready to describe the attack. We choose 2^{41} plaintexts whose third words are equal. Given the ciphertexts, we sort (or hash) them by their third and fourth words, and select pairs which collide at these words. It is expected that about $\frac{(2^{41})^2/2}{2^{32}} = 2^{49}$ pairs are selected.

Each selected pair is subjected to the following analysis, consisting of four phases. In the first phase we analyze the first round. We know the two inputs of the G permutation, and its output difference. This G permutation is keyed by 32 bits, and there are about 2^{16} of the possible subkeys that cause the expected difference. This can be done in 2^{16} steps, by guessing the first two bytes of the subkeys, and computing the other two bytes by differential cryptanalytic techniques. As the subkeys of the first and last rounds are the same, we can peel off the last round for each of the possible subkeys.

We then analyze round 4. We know the input and output differences of the G permutation in round 4. Due to the complementation properties [4][4] of the G permutation, we can assume that the inputs are fixed to some arbitrary pair of values, and find about 2^{16} candidate subkeys corresponding to these values. The complexity of this analysis is 2^{16}. We can then complete all the possible combinations of inputs and subkeys using the complementation properties. The analysis of round 29 is similar. We now observe that the same subkey is used in round 4 and in round 29. The possible subkeys of rounds 4 and 29 are kept efficiently by using the complementation property, and thus we cannot directly search for two equal subkey values. Instead, we observe that the XOR value of the first two subkey bytes with the other two subkey bytes is independent of complementation, and we use this XOR value as the common value which is used to join the two lists of subkeys of both rounds. By a proper complementation we get a list of about 2^{16} tuples of the subkey, the input of round 4 and the output of round 29. The complexity of this analysis is about 2^{16} steps. This list can still be subjected to the complementation property to get all the (about 2^{32}) possible combinations.

The third phase joins the two lists, into a list of about 2^{32} entries of the form $(cv_0, \ldots, cv_5, X_3, X_{30})$ where $cv_0, \ldots, cv_5$ are the six key bytes used in rounds 1, 4, and 29, X_3 is the feedback of the XOR operation in round 3 (i.e., the output of the third G permutation), X_{30} is the feedback in round 30 (i.e., the input of the 30'th G permutation, which is the same in both members of the pair if $cv_0, \ldots, cv_5$ are correct). For each of these values we can now encrypt the first half of round 2 (using cv_4 and cv_5) and decrypt the second half of round 3 (using X_3, cv_0, and cv_1). We can view the second half of round 2 and the first half of round 3 as one permutation, which we call G', which has an additional feedback (the third plaintext word) in its middle. We are left now with only two equalities involving $cv_6, \ldots, cv_9$ which should hold, as we know the input and output of round 30, and we know the two outputs of G'. There is only one solution of $cv_6, \ldots, cv_9$ on average, and given the solution we find a key which encrypts the plaintexts to the input difference of the impossible differential after round 4, and decrypts the ciphertexts to the impossible difference before round 29. Therefore, we find a key which is certainly wrong, and thus should be discarded.

In total we find about 2^{32} such keys during the analysis of each pair. By analyzing 2^{49} pairs selected from the 2^{41} chosen plaintexts, we find a total of $2^{49} \cdot 2^{32} = 2^{81}$ keys, but some of them are found more than once. It is expected that a fraction of $(1-2^{-80})^{2^{81}} = 1/e^2 \approx 1/8$ of the keys are not discarded. These keys are then tested by trial encryptions in the fourth phase.

To complete the description of the attack we should describe two delicate im-

[4] The G permutation of Skipjack has $2^{16}-1$ complementation properties: Let $O = G_K(I)$, and let $d = (d_0, d_1)$ be any 16-bit value. Then $O \oplus d = G_{K \oplus (d_1, d_0, d_1, d_0)}(I \oplus d)$.

plementation details: The first detail describes how to find the subkey $cv_6, \ldots, cv_9$ using one table lookup. The inputs and outputs of G and G' consist of 80 bits, and for each choice of the 80-bit query there is on average only one solution for the subkey. Therefore, we could keep a table of 2^{80} entries, each storing the solution(s) for a specific query. But the size of this table and the time of its precomputation are larger than the complexities we can afford. Instead, we observe that the complementation property of the G permutation [4] enables us to fix one of the input words (say to zero) by XORing the other input, the two outputs, and the proposed subkeys (excluding the intermediate feedback of G') by the original value of this input. We can, therefore, reduce the size of the table to 2^{64}, and the precomputation time to 2^{64} as well. Each entry of the table contains on average one 32-bit subkey. The size of the table can be halved by keeping only the first 16 bits of the subkey, observing that the second half can then be easily computed given the first half.

The second delicate implementation detail is related to the way we keep the list of discarded keys. The simplest way is to keep the list in a table of 2^{80} binary entries whose values are initialized to 0, and are set to 1 when the corresponding keys are discarded. But again, this table is too large (although its initialization and update times are still considerably faster than the rest of the attack). Instead, we observe that we can perform the attack iteratively (while caching the results of phase 2), where in each iteration we analyze only the keys whose first two bytes cv_0 and cv_1 are fixed to the index of the iteration. This modification can be performed easily as the attack guesses these two bytes in its first phase, and each guess leads to independent computations. We thus perform exactly the same attack with a different order of instructions. As the first 16 bits of the keys are now fixed in each iteration, the number of required entries in the table is reduced to 2^{64}.

The complexities of phases 1 and 2 are about 2^{16} for each pair, and $2^{49} \cdot 2^{16} = 2^{65}$ in total for all the pairs. The complexity of phase 3 is as follows: For each pair, and for each value in the joined list, we compute two halves of a G permutation and solve for $cv_6, \ldots, cv_9$ given the inputs and outputs of the third G and of G'. Assuming that this solution costs about one computation of a G permutation, the total complexity of phase 3 is $2^{49} \cdot 2^{32}(2 \cdot \frac{1}{2} + 1) = 2^{82}$ computations of a G permutation, which is equivalent to $2^{82}/31 \approx 2^{77}$ encryptions. The complexity of phase 4 is about $2^{80}/8 = 2^{77}$ encryption. Therefore, the total complexity of the attack is about 2^{78} encryptions, which is four times faster than exhaustive search. The average time complexity of the attack is about 2^{77}, which is also four times faster than the average case of exhaustive search.

An attack on the reduced variant consisting of rounds 2 to 32 requires fewer chosen plaintexts, and the same complexity. Given four structures of 2^{32} chosen plaintexts with words 3 and 4 fixed, we can select the $\frac{4 \cdot (2^{32})^2/2}{2^{16}} = 2^{49}$ required pairs, and apply the same attack to these pairs (exchanging rounds 1 and 32, rounds 2 and 31, etc.). This attack can also be applied as a chosen ciphertext

Rounds	Chosen Plaintexts	Steps
25 (5–29)	2^{38}	2^{27}
26 (4–29)	2^{38}	2^{49}
28 (1–28)	2^{34}	2^{77}
29 (1–29)	2^{34}	2^{77}
30 (1–30)	2^{34}	2^{77}
31 (1–31)	2^{41}	2^{78}
31 (2–32)	2^{34}	2^{78}

Table 1. Complexities of Chosen Plaintext Attacks Against Reduced-Round Skipjack.

attack against the variant consisting of rounds 1 to 31 using 2^{34} chosen ciphertext blocks.

6 Discussion and Conclusions

The best complexities of our attack when applied to reduced-round variants of Skipjack are summarized in Table 1.

This attack cannot be directly used against the full 32 rounds of Skipjack because each pair may discard only about 2^{16} keys. However, the analysis of phases 1 and 2 (which in the case of the full Skipjack also includes the analysis of the last round) cannot be reduced below 2^{32} G computations. Therefore, the complexity of the attack is lower bounded by $2^{16}/32 = 2^{11}$ times the number of discarded keys (instead of being a few times smaller than the number of discarded keys), and thus the time required to eliminate all but the correct key is longer than exhaustive search.

Note that the above attacks against Skipjack are independent of the choice of the G permutation or the F table. Also note that if in addition to the 5-round cycle of the key schedule, Skipjack had 5-round groups of rules (instead of 8-round groups of rules), i.e., had consecutive groups of five rounds of Rule A followed by five rounds of Rule B, followed by five Rule A and five Rule B rounds, etc, then it would have a 27-round impossible differential.

We are aware of several impossible differentials of various blockciphers, such as a 7-round impossible differential of Feal [15,13], 5-round impossible differential of DES [14], 20-round impossible differential of CAST-256 [1], 18-round impossible differential of Khufu [12], and 2.5-round impossible differential of IDEA [10]. In a related paper [5] we use these impossible differentials to cryptanalyze IDEA with up to 4.5 rounds, and to cryptanalyze Khufu with up to 20 rounds. Both attacks analyze more rounds than any other published attack against these ciphers.

There are many modifications and extensions of the ideas presented in this paper. For example, cryptanalysis with impossible differentials can be used with low-probability (rather than zero-probability) differentials, can be used with conditional characteristics [2] (or differentials), and can be combined with linear [11] (rather than differential) cryptanalysis.

Designers of new blockciphers try to show that their schemes are resistant to differential cryptanalysis by providing an upper bound on the probability of characteristics and differentials in their schemes. One of the interesting consequences of the new attack is that even a rigorously proven upper bound of this type is insufficient, and that designers also have to consider lower bounds in order to prove resistance against attacks based on impossible or low-probability differential properties.

References

1. Carlisle M. Adams, *The CAST-256 Encryption Algorithm*, AES submission, available at `http://www.entrust.com/resources/pdf/cast-256.pdf`.
2. Ishai Ben-Aroya, Eli Biham, *Differential Cryptanalysis of Lucifer*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'93, pp. 187–199, 1993.
3. Eli Biham, *Cryptanalysis of Ladder-DES*, proceedings of Fast Software Encryption, Haifa, Lecture Notes in Computer Science, pp. 134–138, 1997.
4. Eli Biham, Alex Biryukov, Orr Dunkelman, Eran Richardson, Adi Shamir, *Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR*, proceedings of Selected Areas in Cryptography, SAC'1998, Lecture Notes in Computer Science,1998.
5. Eli Biham, Alex Biryukov, Adi Shamir, *Miss in the Middle Attack on IDEA Reduced to 4.5 Rounds*, in preparation.
6. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
7. Cipher A. Deavours, Louis Kruh, *Machine Cryptography and Modern Cryptanalysis*, Artech House, 1985.
8. Lars Ramkilde Knudsen, *DEAL - A 128-bit Block Cipher*, AES submission, available at `http://www.ii.uib.no/~larsr/papers/deal.ps`, 1998.
9. Lars Ramkilde Knudsen, Matt Robshaw, David Wagner, private communication, 1998.
10. Xuejia Lai, James L. Massey, Sean Murphy, *Markov Ciphers and Differential Cryptanalysis*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'91, pp. 17–38, 1991.
11. Mitsuru Matsui, *Linear Cryptanalysis Method for DES Cipher*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'93, pp. 386–397, 1993.
12. Ralph C. Merkle, *Fast Software Encryption Functions*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'90, pp. 476–501, 1990.

13. Shoji Miyaguchi, Akira Shiraishi, Akihiro Shimizu, *Fast Data Encryption Algorithm FEAL-8*, Review of electrical communications laboratories, Vol. 36, No. 4, pp. 433–437, 1988.
14. National Bureau of Standards, *Data Encryption Standard*, U.S. Department of Commerce, FIPS pub. 46, January 1977.
15. Akihiro Shimizu, Shoji Miyaguchi, *Fast Data Encryption Algorithm FEAL*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'87, pp. 267–278, 1987.
16. David Wagner, *Further Attacks on 16 Rounds of SkipJack*, private communication, July 1998.