J. Cryptology (2005) 18: 291–311
DOI: 10.1007/s00145-005-0129-3

Journal of
CRYPTOLOGY

Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials*

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

Alex Biryukov
Dept. ESAT/SCD-COSIC, Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
http://www.esat.kuleuven.ac.be/∼abiryuko/

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

Communicated by Bart Preneel

Received 16 August 2001 and revised February 2005
Online publication 22 July 2005

Abstract. In this paper we present a cryptanalytic technique, based on *impossible differentials*. We use it to show that recovering keys of Skipjack reduced from 32 to 31 rounds can be performed faster than exhaustive search. We also describe the Yoyo game (a tool that can be used against reduced-round Skipjack), and other properties of Skipjack.

Key words. Skipjack, Cryptanalysis, Differential cryptanalysis, Impossible differentials, Yoyo game, Adaptive attacks.

1. Introduction

Skipjack [22] is a block cipher with 80-bit keys and 64-bit blocks. It was developed by the NSA for the Clipper chip initiative (including the Capstone chip [21] and the Fortezza PC

* This paper is an extended version of a paper which appeared under the same title at EUROCRYPT '99. The first author is supported by the Israeli Ministry of Science and Technology. During this work Alex Biryukov was with the Applied Mathematics Department, Technion – Israel Institute of Technology, Haifa 32000, Israel.

card), as a member of a family of "Type I" encryption algorithms suitable for protecting all levels of classified data. It was implemented in tamper-resistant hardware, and its structure was kept secret since its introduction in 1993.

To increase confidence in the strength of Skipjack and the Clipper chip initiative, five well-known cryptographers were assigned in 1993 to analyze Skipjack and report their findings [8]. They investigated the strength of Skipjack using differential cryptanalysis [6] and other methods, and concentrated on reviewing NSA's design and evaluation process.

On 24th June 1998 Skipjack was declassified, and its description was made public in the web site of NIST [22]. Immediately after the declassification, two groups of researchers were studying its security simultaneously, and both shared their ideas during the analysis. Our group developed differential and linear cryptanalysis of Skipjack [3]. We analyzed variants of Skipjack with up to 16 rounds with 2^{22} complexity and 2^{22} chosen plaintexts. We also analyzed a slightly modified variant of the full 32-round Skipjack, from which only three XOR operations (out of the 320 XOR operations) are removed. We called this variant *Skipjack-3XOR* (Skipjack minus three XORs). We could attack this variant in less than a million steps using 500 chosen plaintexts. This attack can therefore be carried out on any personal computer in just a few seconds. We also developed the Yoyo game, described in Appendix A of this paper.

In parallel, the other group of researchers, including Knudsen, Robshaw, and Wagner, took a different direction. They used (word-wise) truncated differentials and got the following results [15]: Skipjack reduced to (the first) 16 rounds can be attacked with 2^{17} chosen plaintexts and 2^{34} time of analysis. This attack works even if the subkeys are independent, in which case the same amount of chosen plaintexts is required, but the time of analysis grows to 2^{49}. An attack on Skipjack with the middle 16 rounds requires only three chosen plaintexts and 2^{30} time of analysis. They can even attack Skipjack reduced to (the last) 28 rounds with 2^{77} steps and 2^{41} chosen plaintexts. In addition they used boomerang attacks (a kind of an adaptive chosen plaintext/chosen ciphertext attack) against variants of Skipjack, with which they could distinguish whether a black box cipher performs a 24-round reduced variant of Skipjack, and could find the key of a 25-round reduced variant using $2^{34.5}$ adaptive texts and $2^{61.5}$ time of analysis. Note that in 2001 Granboulan found a problem in some of these attacks, except for 16-round attacks, which still work as described [10], [11].

In addition, it is worth noting that Skipjack can be attacked by the generic time–memory tradeoff approach [12] which requires 2^{80} steps of precomputation and 2^{54} 80-bit words (i.e., 2^{60} bits) of memory, and in which each search for a key requires only 2^{54} steps of computation.

In this paper we devise a new variant of differential cryptanalysis and use it to analyze Skipjack. 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.[1]

[1] Such a basic approach to differential cryptanalysis considers probabilities of differentials averaged over all the keys. However, in some ciphers, considering key-dependent differentials is beneficial to the attacker (see for example [1]). Moreover, one could exploit both high and low probability key-dependent differentials if the wrong pairs would not suggest the correct value of the key, as is demonstrated experimentally in [7].

In contrast, in the new variant of differential cryptanalysis 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*.

We should emphasize that the idea of using impossible events in cryptanalysis is not new. It is well known [9] 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 guessed 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 four-round Feistel cipher using DES as the F function. This cryptanalysis was published in [2], and was based on the fact that collisions cannot be generated by a permutation. A zero probability differential was later used by Knudsen in his description of DEAL [14], a six-round Feistel cipher with 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.

Cryptanalysis with impossible differentials is very powerful against many ciphers with various structures. In this paper we describe an impossible differential of Skipjack [22], [21] 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

1. 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 (using 2^{34} chosen plaintexts and 2^{64} bits of memory),
2. attack shorter variants efficiently (in the case of the 25-round and 26-round variants the complexity is only 2^{38} chosen plaintexts, and 2^{27} and 2^{49} steps, respectively), and
3. distinguish whether a black box applies to a 24-round variant of Skipjack.

In a related paper [4] we describe the application of this type of cryptanalysis to IDEA [16] and to Khufu [18]. These attacks improved the best known attacks on these schemes.

We also present a new cryptographic tool, which we call the *Yoyo game*, applied to Skipjack reduced to 16 rounds. This tool can be used to identify pairs satisfying a certain property, and be used as a tool for attacking Skipjack reduced to 16 rounds using only 2^{14} adaptive chosen plaintexts and ciphertexts and 2^{14} steps of analysis. This tool can also be used as a distinguisher to decide whether a given black box contains this variant of Skipjack.

Table 1 summarizes the attacks against Skipjack.

Table 1. Summary of attacks against reduced-round Skipjack.

Ref.	Rounds	#Texts	Steps	Memory	
[12]	Any	2	2^{54}	2^{54}	Provided 2^{80} steps of precomutation are performed
[15]	16 (1–16)	2^{17}	2^{34}		
	16 (9–24)	3	2^{30}		
	28 (5–32)*	2^{41}	2^{77}		
	24 (5–28)*	2^{25}	2^{25}		Boomerang; distinguishing
	25 (4–28)*	$2^{34.5}$	$2^{61.5}$		Boomerang
[13]	22(1–22)	2^{49}	2^{44}	2^{49} texts	Multiset/saturation attack
	27(1–27)	2^{50}	$2^{76.6}$	2^{50} texts	
[3]	16 (1–16)	2^{22}	2^{22}	–	
	16 (1–16)	2^{14}	2^{16}	–	Yoyo game; distinguishing
	Skipjack-3XOR	2^{9}	2^{20}	–	32 rounds
This	25 (5–29)	2^{38}	2^{27}	–	
paper	26 (4–29)	2^{38}	2^{49}	–	
	28 (1–28)	2^{34}	2^{77}	2^{64} bits	
	29 (1–29)	2^{34}	2^{77}	2^{64} bits	
	30 (1–30)	2^{34}	2^{77}	2^{64} bits	
	31 (1–31)	2^{41}	2^{78}	2^{64} bits	
	31 (2–32)	2^{34}	2^{78}	2^{64} bits	

*Does not work according to [10].

The paper is organized as follows: A description of Skipjack is given in Section 2. A 24-round impossible differential of Skipjack is described in Section 3. In Section 4 we use this impossible differential for a distinguishing attack on 24-round Skipjack. In Section 5 we use it to attack Skipjack reduced to 25 and to 26 rounds, and in Section 6 we describe our main attack against Skipjack reduced to 31 rounds. Finally, in Section 7 we discuss why the attack is not directly applicable to the full 32-round Skipjack, and summarize the paper. In the appendices we describe the Yoyo game, an automated approach for finding impossible differentials, complementation properties of the G permutation, and analysis of modified variants of Skipjack.

2. Description of Skipjack

Skipjack is an iterated block cipher 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 an 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 Rules A and B are given in Fig. 1, where the round number ($k+1$, also called the round counter in the original description of Skipjack) is in the range 1–32, k is the round number minus one (in the range 0–31), G is a four-round Feistel permutation whose F function is based on an (8×8)-bit S box, called *F Table*, and each round of G is keyed by eight bits of the key. See Fig. 2 for an outline of the G permutation, in which the cv's are the four bytes of subkey.

Rule A	Rule B
$w_1^{k+1} = G^k(w_1^k, \text{subkey}^k) \oplus w_4^k \oplus (k+1)$	$w_1^{k+1} = w_4^k$
$w_2^{k+1} = G^k(w_1^k, \text{subkey}^k)$	$w_2^{k+1} = G^k(w_1^k, \text{subkey}^k)$
$w_3^{k+1} = w_2^k$	$w_3^{k+1} = w_1^k \oplus w_2^k \oplus (k+1)$
$w_4^{k+1} = w_3^k$	$w_4^{k+1} = w_3^k$

Fig. 1. Rules A and B.

The description becomes simpler (and the software implementation becomes more efficient) if we unroll the rounds, and keep the four elements in the shift register stationary. In this form the code is simply a sequence of alternate G operations and XOR operations of cyclically adjacent elements. In this representation the main difference between Rules A and B is the direction in which the adjacent elements are XORed (left to right or right to left).

The XOR operations of Rules A and B after round 8 and after round 24 (on the borders between Rules A and B) are consecutive without application of the G permutation in between. In the unrolled description these XORs (in rounds 8–9) are of the form

$$\begin{aligned} w_2^8 &= G(w_2^7, \text{subkey}^7) && \texttt{(Rule A)}, \\ w_1^8 &= w_1^7 \oplus w_2^8 \oplus 8, \\ w_2^9 &= w_1^8 \oplus w_2^8 \oplus 9 && \texttt{(Rule B)}, \\ w_1^9 &= G(w_1^8, \text{subkey}^8), \end{aligned}$$

which is equivalent to exchanging the words w_1 and w_2, and leaving w_2 as the original $w_1 \oplus 1$:

$$\begin{aligned} w_2 &= G(w_2, \text{subkey}^7), \\ &\text{exchange } w_1 \text{ and } w_2, \\ w_1 &= w_1 \oplus w_2 \oplus 8, \\ w_2 &= w_2 \oplus 1, \\ w_1 &= G(w_1, \text{subkey}^8), \end{aligned}$$

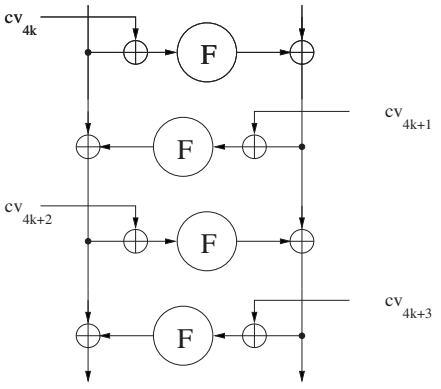

Fig. 2. Outline of the G permutation.

or even

$$\begin{aligned} w_1^8 &= G(w_2^7, \text{subkey}^7) \oplus w_1^7 \oplus 8, \\ w_1^9 &= G(w_1^8, \text{subkey}^8), \\ w_2^9 &= w_1^7 \oplus 1 \end{aligned}$$

(the same situation occurs after round 24 with the round numbers 8 and 9 replaced by 24 and 25). Figure 3 describes this representation of Skipjack (only the first 16 rounds out of the 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 Rules A and B rounds.

Also, on the border between Rules B and A (after round 16), there are two parallel applications of the G permutation on two different words, with no other linear mixing in between.

Note that Rule A mixes the output of the G permutation into the input of the next G permutation, while Rule B mixes the input of a G permutation into the output of the previous G permutation (similarly in decryption of Rule A), and thus during encryption Rule B rounds add little to the avalanche effect, and during decryption Rule A rounds add little to the avalanche effect.

2.1. *The Key Schedule*

Skipjack keys contain ten bytes. In each round four consecutive bytes of the key are used as the subkey. In the first round the first four bytes are used, and in each successive round, the next four bytes (cyclically) are used.

As a result, the key schedule has the following properties: The subkeys are cyclic in the sense that the same set of four bytes of the subkeys (entering a single G permutation) are repeated every five rounds, and there are only five such sets. In addition, the key bytes are divided into two sets: the even bytes and the odd bytes. The even bytes always enter the even rounds of the G permutation, while the odd bytes always enter the odd rounds of the G permutation.

2.2. *Decryption*

Decryption is performed by applying the inverse of all operations from the last round to the first. We observe that decryption can be performed using the same procedure as encryption with minor modifications. These modifications are

1. reordering the key bytes to

$$K^* = (cv_7, cv_6, cv_5, cv_4, cv_3, cv_2, cv_1, cv_0, cv_9, cv_8),$$

2. reversing the order of the round counters $(k + 1)$ mixed into the data, and then
3. encrypting the reordered ciphertext

$$C^* = (cb_3, cb_2, cb_1, cb_0, cb_7, cb_6, cb_5, cb_4)$$

gives the reordered plaintext

$$P^* = (pb_3, pb_2, pb_1, pb_0, pb_7, pb_6, pb_5, pb_4).$$

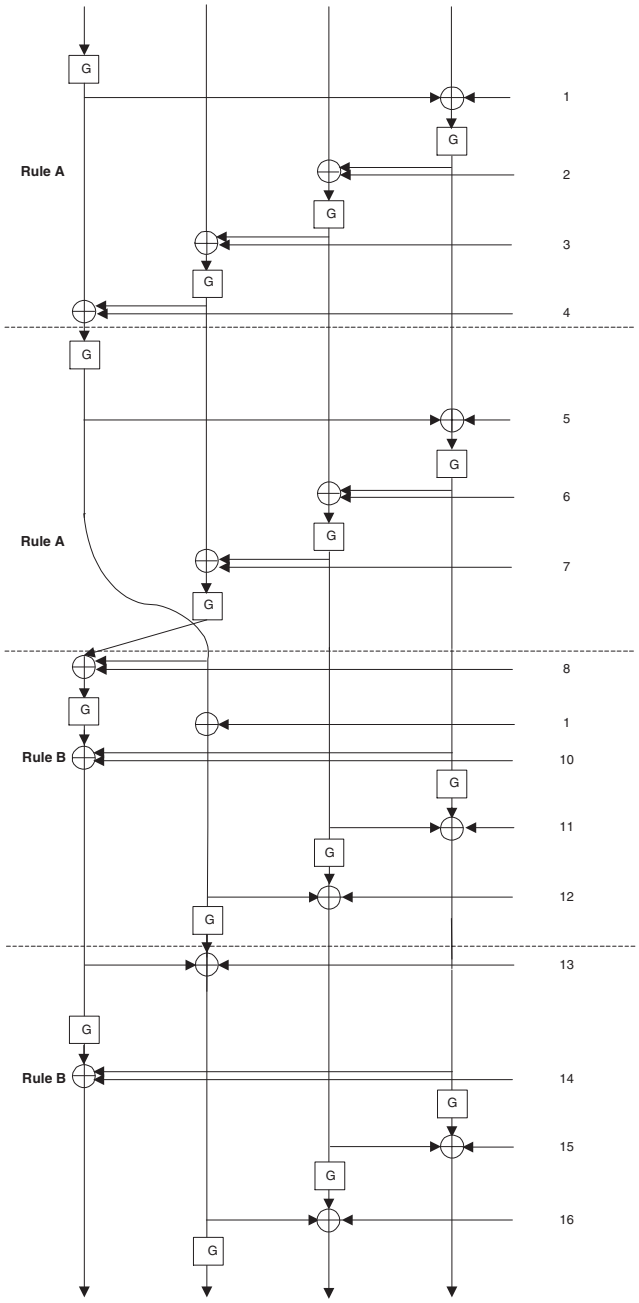

Fig. 3. The first 16 rounds of Skipjack (the next 16 rounds are identical except for the round counters).

3. A 24-Round Impossible Differential

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.

The reason that this differential has probability 0 can be explained by a *miss in the middle* approach, where two 12-round differentials with probability 1 evolve from both ends of the 24 rounds towards the middle, but they miss to agree on a common difference in the middle:

1. As Wagner observed in [25], 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. This differential is outlined in Fig. 4.
2. 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.

As outlined in Fig. 5, these two differentials cannot be combined. Any pair with difference $(0, a, 0, 0)$ after round 4 and difference $(b, 0, 0, 0)$ after round 28 must have a difference 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.

4. Distinguishing Attacks

One application of this differential may be to test whether an encryption black box is a 24-round Skipjack (from round 5 to round 28). It only requires to feed the black box with $2^{48}\alpha$ pairs (for some α) with differences of the form $(0, a, 0, 0)$, and to test 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 another 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.

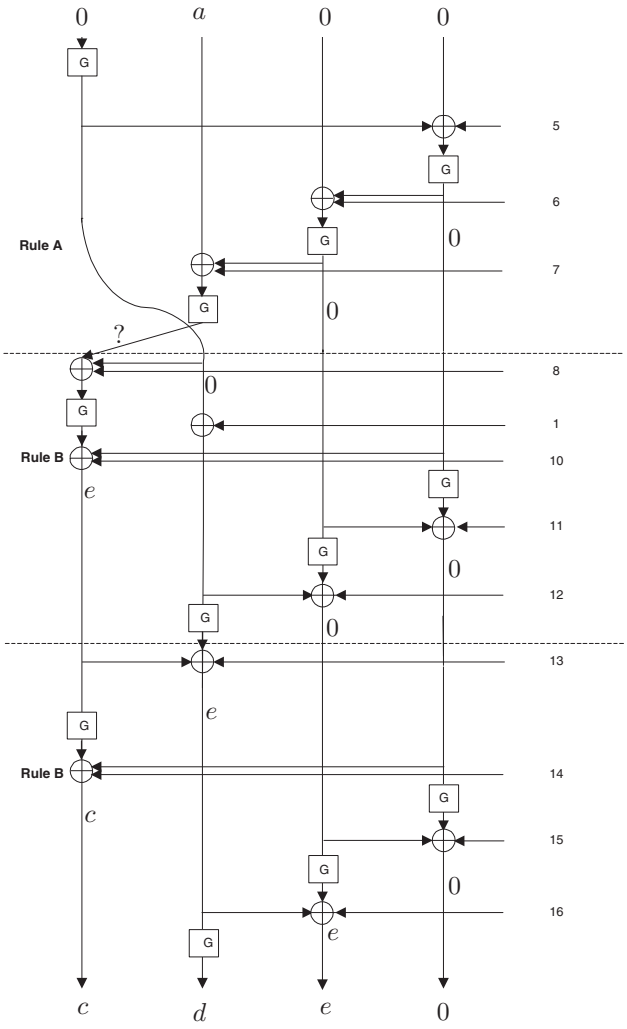

Fig. 4. The 12-round differential of rounds 5–12 with probability 1, the differences are marked on the figure, where a, c, d, e, and ? denote non-zero differences.

5. Attack on Skipjack Reduced to 25–26 Rounds

In this section we describe the simplest (key-recovery) 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 contain 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

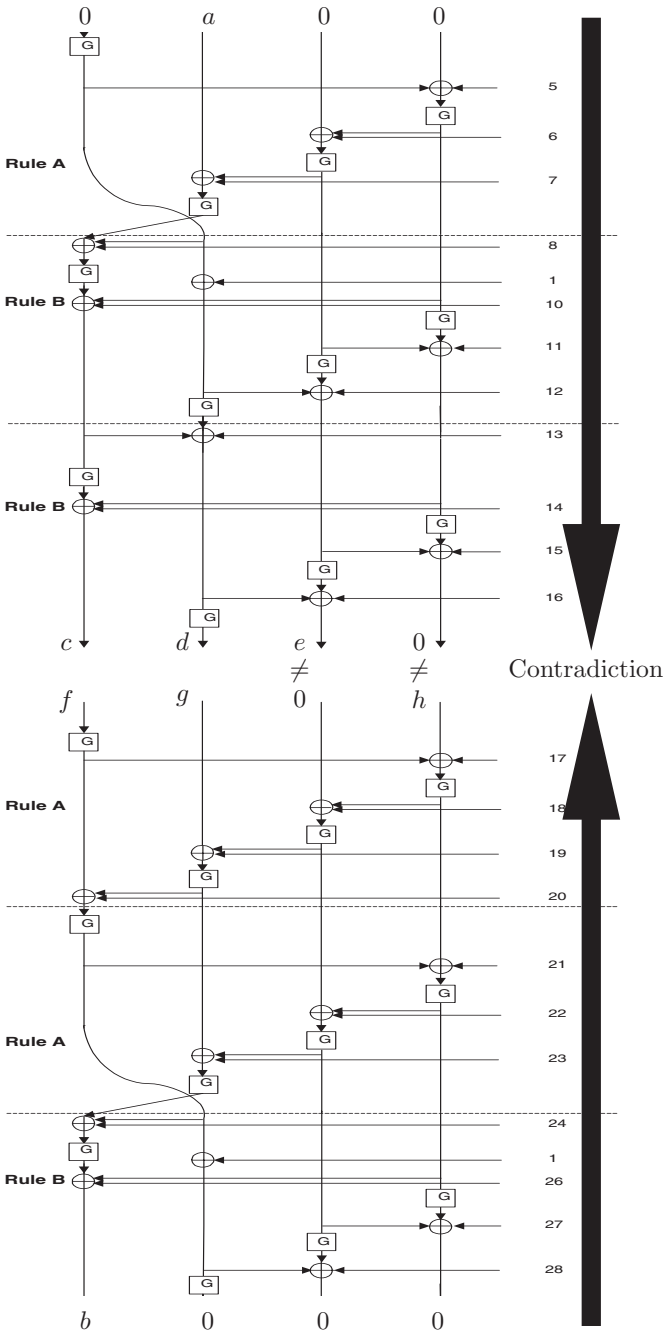

Fig. 5. Miss in the middle: the 24-round impossible differential.

contain 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 suggests 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 suggest 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 suggests 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, or by more efficient auxiliary techniques.

Note that in this and in the following key recovery attacks we assume that discarded keys are distributed uniformly at random. This assumption is reasonable since G is a four-round Feistel construction with four independent key bytes and we may assume that input/output constraints on G are uniformly distributed for different pairs. Note also that the standard S/N reasoning in differential cryptanalysis implies a similar assumption. Another important assumption is that ciphertexts do follow the difference of type $(x, y, 0, 0)$, $x \neq 0$, $y \neq 0$ with probability 2^{-32}. This assumption could be invalidated by the presence of impossible or low probability differentials. In the attacks presented here and in the following sections this is however not the case. Moreover, the attacker would be eager to exploit such properties of a cipher since they would extend the number of rounds of the distinguisher and thus would allow to break even more rounds.

6. Attack on 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.

6.1. *Preliminaries*

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.

6.2. *General Structure of the Attack*

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 suggested by the impossible differential do hold, and discard all keys which confirm this false assumption.

6.3. *The Attack*

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 $((2^{41})^2/2)/2^{32} = 2^{49}$ 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. The subkeys used in this round can be recovered within 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 of the G permutation (see Appendix C and [3]), 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 (16-bit) feedback of the XOR operation in round 3 (i.e., the output of the third G permutation), X_{30} is the (16-bit) feedback in round 30 (i.e., the input of the 30th 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 $(1 - 2^{-80})^{2^{81}} = 1/e^2 \approx \frac{1}{8}$ of the keys are not discarded. These keys are then tested by trial encryptions in the fourth phase.

6.4. *Implementation Details*

To complete the description of the attack we should describe two delicate implementation 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. However, 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 (see Appendix C and [3]) enables us to fix one of the input words (say to zero) by XORing the other input, the two outputs, and the suggested 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. However, 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} bits.

6.5. *Complexity*

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.

6.6. *Cryptanalysis of the Last* 31 *Rounds of Skipjack*

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 $(4 \cdot (2^{32})^2/2)/2^{16} = 2^{49}$ 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 attack against the variant consisting of rounds 1–31 using 2^{34} chosen ciphertext blocks.

7. Discussion and Conclusions

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 F table, and that the attacks on the 25-round and 26-round variants are also independent of the choice of the G permutation.

Also note that the order of Rules A and B is important: If in addition to the five-round cycle of the key schedule, Skipjack had five-round groups of rules (instead of eight-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 block ciphers, such as a nine-round impossible differential of Feal [24], [19], a seven-round impossible differential of DES [20], an 18-round impossible differential of Khufu [18], and a 2.5-round impossible differential of IDEA [16]. In a related paper [4] 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 previously 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 [1] (or differentials), and can be combined with linear [17] (rather than differential) cryptanalysis.

Designers of new block ciphers 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.

Acknowledgments

We are grateful to David Wagner, Lars Knudsen, and Matt Robshaw for sharing various beautiful observations and results with us, and to the anonymous referees of both EUROCRYPT '99 and the *Journal of Cryptology*. We are also grateful to Rivka Zur, the Technion CS secretary, for preparing Figure 3.

Appendix A. A New Cryptographic Tool: The Yoyo Game

Consider the first 16 rounds of Skipjack, and consider pairs of plaintexts $P = (w_1, w_2, w_3, w_4)$ and $P^* = (w_1^*, w_2^*, w_3^*, w_4^*)$ whose partial encryptions differ only in the second

word in the input of round 5 (we refer to it as the *property* from now on). As this word does not affect any other word until it becomes word 1 in round 12, the other three words have difference zero between rounds 5 and 12.

We next observe that given a pair with such a property, we can exchange the second words of the plaintexts (which cannot be equal if the property holds), and the new pair of plaintexts (w_1, w_2^*, w_3, w_4) and $(w_1^*, w_2, w_3^*, w_4^*)$ still satisfies the property, i.e., differs only in the second word in the input of round 5. Given the ciphertexts we can carry out a similar operation of exchanging words 1.

The Yoyo game starts by choosing an arbitrary pair of distinct plaintexts P_0 and P_0^*. The plaintexts are encrypted to C_0 and C_0^*. We exchange the first words of the two ciphertexts as described above, receiving C_1 and C_1^*, and decrypt them to get P_1, P_1^*. Now we exchange the second words of the plaintexts, receiving P_2 and P_2^*, and encrypt them to get C_2 and C_2^*. The Yoyo game repeats this forever.

In this game, whenever we start with a pair of plaintexts which satisfies the property, all the resultant pairs of encryptions must also satisfy the property, and if we start with a pair of plaintexts which does not satisfy the property, all the resultant encryptions cannot satisfy it.

It is easy to identify whether the pairs in a Yoyo game satisfy the above property, by verifying whether some of the pairs achieved in the game have a non-zero difference in the third word of the plaintexts or in the fourth word of the ciphertexts. If one of these differences is non-zero, the pair cannot satisfy the property. On the other hand, if the pair does not satisfy the property, there is only a probability of 2^{-16} that the next pair in the game has difference zero, and thus it is possible to stop games in which the property is not satisfied after only a few steps. If the game is not stopped within a few steps, we conclude with overwhelming probability that the property is satisfied.

This game can be used for several purposes. The first is to identify whether a given pair satisfies the above property, and to generate many additional pairs satisfying the property.

This can be used to attack Skipjack reduced to 16 rounds in just 2^{14} steps. For the sake of simplicity, we describe a suboptimal implementation with complexity 2^{17}. In this version we choose 2^{17} plaintexts whose third word is fixed. This set of plaintexts defines about 2^{33} possible pairs, of which about 2^{17} candidate pairs have difference zero in the fourth word of the ciphertexts, and of which about one or two pairs are expected to satisfy the property. Up to this point, this attack is similar to Wagner's attack on 16-round Skipjack [25]. We then use the Yoyo game to reduce the complexity of analysis considerably. We play the game for each of the 2^{17} candidate pairs, and within a few steps of the game discard all the pairs which do not satisfy the property. We are left with one pair which satisfies the property, and with several additional pairs generated during the Yoyo game which also satisfy the property. Using two or three of these pairs, we can analyze the last round of the cipher and find the unique subkey of the last round that satisfies all the requirements with complexity about 2^{16}. The rest of the key bytes can be found by similar techniques.

This game can also be used as a distinguisher which can test whether an unknown encryption algorithm (given as an oracle) performs Skipjack reduced to 16 rounds.

The above Yoyo game keeps three words with difference zero in each pair. We note that there is another (less useful) Yoyo game for Skipjack reduced to 14 rounds (specif-

ically, rounds 2–15), which keeps only one word with difference zero. Consider pairs of encryptions $P = (w_1, w_2, w_3, w_4)$ and $P^* = (w_1^*, w_2^*, w_3^*, w_4^*)$ which have the same data at the leftmost word in the input of round 5. As this word is not affected by any other word until it becomes word 2 in round 12, we can conclude that both encryptions have the same data in word 2 after round 12. Given a pair with such an equality in the data, we can exchange the first word of the plaintexts, and the new pair of plaintexts (w_1^*, w_2, w_3, w_4) and $(w_1, w_2^*, w_3^*, w_4^*)$ still has the same property of equality at the input of round 5. Moreover, if the first words of the plaintexts are equal (i.e., $w_1 = w_1^*$ and thus exchanging them does nothing) we can exchange the second words (w_2 with w_2^*) and get the same property. If they are also equal, we can exchange w_3 with w_3^* and get the same property. If they are also equal, we exchange w_4 with w_4^*. However, if the property holds, this last case is impossible, as at least two words of the two plaintexts must be different. Given the ciphertexts we can carry out a similar operation of exchanging words 2. If words 2 are equal, exchange words 1, then words 4, and then words 3. Also in this case a difference of only one word ensures that the property is not satisfied. This Yoyo game is similar to the previous game, except for its modified exchange process, and it behaves similarly with respect to the new difference property.

Appendix B. *Shrinking*: An Automated Technique for Finding Global Impossible Differentials

In Section 3 we used the miss in the middle approach to find the 24-round impossible differential of Skipjack. In this appendix we describe an automated approach for finding all the impossible differentials which are based on the global structure of the cipher. The simplest way to automate the search is to encrypt many pairs of plaintexts under various keys, and to conclude that every differential suggested by the encrypted plaintexts (i.e., any differential formed by a plaintext difference and the corresponding ciphertext difference) is not an impossible differential. Therefore, by elimination, only differentials that never occur in our trials may be impossible.

The main problem is that the space of differentials is too large. The problem can be greatly simplified when considering word-wise truncated differentials whose differences distinguish only between zero and arbitrary non-zero differences in the various words (e.g., Skipjack divides the blocks into four words, and thus there are only 16 possible truncated plaintext differences, and 16 possible truncated ciphertext differences, yielding 256 truncated differentials). By selecting various plaintext pairs and computing the ciphertext differences, we can easily discard most differentials which are not impossible. However, when long blocks are divided into many small words, we may never encounter an input pair whose outputs are almost identical, except for a single word.

To overcome this problem we analyze scaled down variants of the cipher, which preserve its global structure but change its local details (including the size of words and the definition of the various functions and permutations). In many cases, including the impossible differential used against Skipjack in this paper, the particular implementation of the G permutation, the F table, and the key schedule do not affect the impossible differentials. In such cases we can replace the local operations in the cipher by other

operations, maintaining the global structure. Moreover, we can also reduce the word size to a smaller word size, together with reducing the size of the local operations without affecting the impossible differentials. We therefore replace the word size by a few bits (typically three, since any invertible function with fewer bits is affine), and replace the large functions by appropriate smaller functions.[2] Impossible differentials resulting from the global structure of the cipher remain impossible even in the scaled down variant. As the block size of the new variant is small (e.g., 12 bits in the case of Skipjack), we can easily encrypt all the 2^{12} plaintexts and calculate all their differences (by exhaustive computation of all the 2^{23} pairs of plaintexts and ciphertexts). By repeating this process for several random independent choices of the local functions, and taking the intersection of the resulting impossible differentials, we can get with high probability all the impossible differentials which are a consequence of the global structure of the cipher.[3] We call this technique *shrinking*.

Using this approach we searched for the word-wise truncated impossible differentials of Skipjack with various numbers of rounds. We found a large number of impossible differentials with fewer than 24 rounds (some of them with more than one non-zero word difference in the plaintext or the ciphertext), and confirmed that the longest impossible differential based on the global structure of Skipjack has 24 rounds. The most notable shorter impossible differentials of Skipjack are (1) the two 23-round impossible differentials (rounds 5–27) which are $(0, a, 0, 0) \not\rightarrow (b, 0, 0, 0)$ and $(0, a, 0, 0) \not\rightarrow (0, b, 0, 0)$ (where a and b are non-zero), and (2) the two 22-round impossible differentials (rounds 5–26) which are $(0, a, 0, 0) \not\rightarrow (0, b, 0, 0)$, and the more useful $(0, a, 0, 0) \not\rightarrow (x, 0, y, 0)$, where x and y can have any value.[4]

Appendix C. Complementation Properties of the G Permutation

The G permutation has $2^{16} - 1$ complementation properties: let $G_{K0,K1,K2,K3}(x1, x2) = (y1, y2)$, where $K0$, $K1$, $K2$, $K3$, $x1$, $x2$, $y1$, $y2$ are all byte values, and let $d1$, $d2$ be two byte values. Then

$$G_{K0\oplus d1,K1\oplus d2,K2\oplus d1,K3\oplus d2}(x1 \oplus d2, x2 \oplus d1) = (y1 \oplus d2, y2 \oplus d1).$$

G has exactly one fixpoint for every subkey (this was identified by Gifford, and described in sci.crypt). Moreover, we observed that for every key and every value v of the form $(0, b)$ or $(b, 0)$ where 0 is a zero byte and b is an arbitrary byte value, G has exactly one value x for which $G(x) = x \oplus v$. It is unknown whether this property can aid in the analysis of Skipjack.

---

[2] The new functions should preserve the main character of the original functions. For example, large permutations should be replaced by smaller permutations, linear functions by smaller linear functions, etc.

[3] This technique can also find word-wise truncated differentials with probability 1 which are based on the global structure of the cipher.

[4] In [23] it is claimed that these differentials (as given also in [5]) are incorrect. Unfortunately this is their mistake in interpreting the paper: we use the cipher as referred to in Fig. 3, i.e., without the rotation of words, while [23] finds the same differentials but lists them in the original order of words (with rotations). Notice that only a shift by one word is between our and their differentials in round 27, and two words in round 26.

Appendix D. Two Attacks on a Variant without the Round Counters

The mixings with the round numbers are often used to protect against related key attacks. The following property demonstrates that this mixing is essential in Skipjack. If these mixings are removed, then given a plaintext

$$P = (pb_0, pb_1, \ldots, pb_7),$$

a key

$$K = (cv_0, \ldots, cv_9),$$

and a ciphertext

$$C = (cb_0, \ldots, cb_7),$$

such that

$$C = \text{Skipjack}_K(P),$$

then decryption can be performed using encryption by

$$P^* = \text{Skipjack}_{K^*}(C^*),$$

where

$$\begin{aligned} K^* &= (cv_7, cv_6, \ldots, cv_0, cv_9, cv_8), \\ P^* &= (pb_3, pb_2, pb_1, pb_0, pb_7, pb_6, pb_5, pb_4), \end{aligned}$$

and

$$C^* = (cb_3, cb_2, cb_1, cb_0, cb_7, cb_6, cb_5, cb_4).$$

This property could be used to reduce the complexity of exhaustive search of this Skipjack variant by a factor of almost 2 (26% of the key space rather than 50% on average) in a similar way to the complementation property of DES: Given the encrypted ciphertext $C1$ of some plaintext P, and the decrypted plaintext $C2$ of the related P^* under the same unknown key, perform trial encryptions with 60% of the keys K (three keys of each cycle of five keys of the rotation by two key bytes operations; efficient implementations first try two keys of each cycle, and only if all of them fail, they try the third keys of the cycles). For each of these keys compare the ciphertext with $C1$, and with $C2^*$ (i.e., $C2$ in which the bytes are reordered as above). If the comparison fails, the unknown key is neither K nor K^*. If it succeeds, we make two or three trial encryptions, and in case they succeed we found the key.

Moreover, the same property causes a class of 2^{40} weak keys for the version of Skipjack without the round counters. These are the keys for which the following relations,

$$cv_0 = cv_7, \quad cv_1 = cv_6, \quad cv_2 = cv_5, \quad cv_3 = cv_4, \quad cv_8 = cv_9,$$

hold simultaneously. For this class of keys, encryption and decryption are the same up to a reordering of the plaintext and ciphertext bytes. Thus, a membership test for this class

of weak keys requires two chosen plaintext queries of two plaintext blocks P and P^* (whose byte reordering relation is as mentioned above). If the obtained ciphertexts are related by byte-reordering, then it is highly probable that the key belongs to the weak key class. The key itself can then be discovered in 2^{40} steps just by searching exhaustively through all the keys in the described class. The same attack can also be performed with a single "self-related" text (!) for which $P = P^*$ (e.g., $P = P^* = 0$). For the weak key the resultant ciphertext should also follow $C = C^*$, which is a 40-bit condition to filter the wrong keys. Working with two such texts would impose an 80-bit condition, thus leaving almost no false alarms. Note that given about 2^{33} known plaintexts we can find two self-related texts with high probability.

References

[1] Ishai Ben-Aroya, Eli Biham, Differential Cryptanalysis of Lucifer, *Advances in Cryptology*, *Proceedings of CRYPTO* '93, Lecture Notes in Computer Science 773, pp. 187–199, Springer-Verlag, Berlin, 1993.

[2] Eli Biham, Cryptanalysis of Ladder-DES, *Proceedings of Fast Software Encryption*, Lecture Notes in Computer Science 1267, pp. 134–138, Springer-Verlag, Berlin, 1997.

[3] 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'* 98, Lecture Notes in Computer Science 1556, pp. 362–375, Springer-Verlag, Berlin, 1998.

[4] Eli Biham, Alex Biryukov, Adi Shamir, Miss in the Middle Attacks on IDEA and Khufu, *Proceedings of Fast Software Encryption – FSE* '99, Lecture Notes in Computer Science 1636, pp. 124–138, Springer-Verlag, Berlin, 1999.

[5] Eli Biham, Alex Biryukov, Adi Shamir, Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials, *Advances in Cryptology*, *Proceedings of EUROCRYPT* '99, Lecture Notes in Computer Science 1592, pp. 12–23, Springer-Verlag, Berlin, 1999.

[6] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, New York, 1993.

[7] Johan Borst, Lars Ramkilde Knudsen, Vincent Rijmen, Two Attacks on Reduced IDEA, *Advances in Cryptology*, *Proceedings of EUROCRYPT* '97, Lecture Notes in Computer Science 1233, pp. 1–13, Springer-Verlag, Berlin, 1997.

[8] Ernest F. Brickell, Dorothy E. Denning, Stephen T. Kent, David P. Maher, Walter Tuchman, SKIPJACK Review, Interim Report, The SKIPJACK Algorithm, July 28, 1993. Available at `http://www.austinlinks.com/Crypto/skipjack-review.html`.

[9] Cipher A. Deavours, Louis Kruh, *Machine Cryptography and Modern Cryptanalysis*, Artech House, Norwood, MA, 1985.

[10] Louis Granboulan, Flaws in Differential Cryptanalysis of Skipjack, *Proceedings of Fast Software Encryption — FSE* 2001, Lecture Notes in Computer Science 2355, pp. 328–335, Springer-Verlag, Berlin, 2001.

[11] Louis Granboulan, Lars Knudsen, David Wagner, Private communication, 2003.

[12] M. E. Hellman, A Cryptanalytic Time–Memory Tradeoff, *IEEE Transactions on Information Theory*, Vol. 26, No. 4, pp. 401–406, July 1980.

[13] Kyungdeok Hwang, Wonil Lee, Sungjae Lee, Sangjin Lee, Jongin Lim, Saturation Attacks on Reduced Round Skipjack, *Proceedings of Fast Software Encryption* — FSE 2002, Lecture Notes in Computer Science 2365, pp. 100–111, Springer-Verlag, Berlin, 2002.

[14] Lars Ramkilde Knudsen, DEAL - A 128-bit Block Cipher, AES submission, 1998.

[15] Lars Ramkilde Knudsen, Matt J.B. Robshaw, David Wagner, Truncated Differentials and Skipjack, *Advances in Cryptology*, *Proceedings of CRYPTO* '99, Lecture Notes in Computer Science 1666, pp. 165–180, Springer-Verlag, Berlin, 1999.

[16] Xuejia Lai, James L. Massey, Sean Murphy, Markov Ciphers and Differential Cryptanalysis, *Advances in Cryptology*, *Proceedings of EUROCRYPT* '91, Lecture Notes in Computer Science 547, pp. 17–38, Springer-Verlag, Berlin, 1991.

[17] Mitsuru Matsui, Linear Cryptanalysis Method for DES Cipher, *Advances in Cryptology*, *Proceedings of EUROCRYPT* '93, Lecture Notes in Computer Science 765, pp. 386–397, Springer-Verlag, Berlin, 1994.

[18] Ralph C. Merkle, Fast Software Encryption Functions, *Advances in Cryptology*, *Proceedings of CRYPTO* '90, Lecture Notes in Computer Science 537, pp. 476–501, Springer-Verlag, Berlin, 1991.

[19] 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.

[20] National Bureau of Standards, *Data Encryption Standard*, U.S. Department of Commerce, FIPS publication 46, January 1977.

[21] National Security Agency, Capstone (MYK-80) Specifications (U), R21 Informal Technical Report, R21-TECH-30-95, 14 August 1995, TOP SECRET, Not Releasable to Foreign Nationals, Declassified in 1999.

[22] National Security Agency, Skipjack and KEA Algorithm Specifications, Version 2.0, 29 May 1998. Available at the National Institute of Standards and Technology's web page, `http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack-kea.htm`.

[23] Ben Reichardt, David Wagner, Markov Truncated Differential Cryptanalysis of Skipjack, *Proceedings of Selected Areas in Cryptography*, *SAC* '2002, Lecture Notes in Computer Science 2595, pp. 110–128, Springer-Verlag, Berlin, 2002.

[24] Akihiro Shimizu, Shoji Miyaguchi, Fast Data Encryption Algorithm FEAL, *Advances in Cryptology*, *Proceedings of EUROCRYPT* '87, Lecture Notes in Computer Science 304, pp. 267–278, Springer-Verlag, Berlin, 1988.

[25] David Wagner, Further Attacks on 16 Rounds of Skipjack, Private communication, July 1998.