J. Cryptology (1994) 7: 229–246

Journal of
CRYPTOLOGY

New Types of Cryptanalytic Attacks Using Related Keys*

Eli Biham
Computer Science Department, Technion—Israel Institute of Technology,
Haifa 32000, Israel

Communicated by Don Coppersmith

Received 8 October 1992 and revised 21 April 1993

Abstract. In this paper we study the influence of key-scheduling algorithms on the strength of blockciphers. We show that the key-scheduling algorithms of many blockciphers inherit obvious relationships between keys, and use these key relations to attack the blockciphers. Two new types of attacks are described: New chosen plaintext reductions of the complexity of exhaustive search attacks (and the faster variants based on complementation properties), and new low-complexity chosen key attacks. These attacks are independent of the number of rounds of the cryptosystems and of the details of the F-function and may have very small complexities. These attacks show that the key-scheduling algorithm should be carefully designed and that its structure should not be too simple. These attacks are applicable to both variants of LOKI and to Lucifer. DES is not vulnerable to the related keys attacks since the shift pattern in the key-scheduling algorithm is not the same in all the rounds.

Key words. Key-scheduling algorithm, DES-like cryptosystems, Chosen key attacks, Chosen plaintext attacks, LOKI, Data Encryption Standard.

1. Introduction

In this paper we describe new types of attacks on blockciphers: chosen plaintext reductions of the complexity of exhaustive search and chosen key attacks in which only the relations between pairs of related keys are chosen by the attacker, who does not know the keys themselves. The chosen plaintext attacks reduce the complexity of exhaustive search and the complexity of the faster chosen plaintext attacks based on complementation properties by a factor of three. The chosen key attacks have very low complexities; however, they can be used only whenever the attacker can choose the relationships between unknown keys and wish to know the keys themselves.

These attacks are based on the observation that in many blockciphers we can view the key-scheduling algorithm as a set of algorithms, each of which extracts

* This research was supported by the fund for the promotion of research at the Technion.

one particular subkey from the subkeys of the previous few rounds. If all the algorithms of extracting the subkeys of the various rounds are the same, then given a key we can shift all the subkeys one round backward and get a new set of valid subkeys which can be derived from some other key. We call these keys *related keys*.

An interesting feature of the attacks based on related keys is that they are independent of the number of rounds of the attacked cryptosystem. These attacks are applicable to both variants of LOKI [5], [4] and to Lucifer [17]. Nevertheless, they are not applicable to DES [15] due to the observation that the number of shifts of the key registers (C and D) in the key-scheduling algorithm is not the same in all the rounds. However, if the shifts by one bit in the key scheduling of DES were replaced by shifts by two bits, DES would have been vulnerable to this kind of attack as well.

Another potential application of related keys is to analyze hash functions (either hash functions based on blockciphers or general hash functions). It may be possible in such functions to choose the message in a way that the related keys property suggest an additional message with the same hash value. Currently, we are not aware of such a particular application to hash functions, but designers of hash functions should be careful to design their functions so that they are immune to this weakness.

The results of the attacks are as follows: The complexity of a chosen plaintext attack on LOKI89 is about $1.5 \cdot 2^{54}$, which is almost three times faster than previously reported chosen plaintext attacks. The chosen key chosen plaintext attack takes a few seconds on a personal computer and its complexity is about 2^{17}, and the complexity of the chosen key known plaintext attack is about 2^{32}. The corresponding complexities of the attacks on the newer LOKI91 are $1.375 \cdot 2^{61}$, 2^{32}, and 2^{48}, respectively. The complexity of the chosen key chosen plaintext attack on Lucifer is about 2^{33}. The DES, the IDEA cipher [12], [13], and the FEAL cipher [16], [14] are not vulnerable to these attacks.

Recently, Knudsen [10] independently found the basic concept of the chosen plaintext related keys attacks and applied it to LOKI91. However, this attack (whose complexity is $1.07 \cdot 2^{62}$) is still 50% slower than the corresponding attack we present in this paper.

2. Description of LOKI89 and LOKI91

LOKI is a family of blockciphers with two variants: the original LOKI cipher, which was renamed LOKI89 [5], and the newer variant LOKI91 [4]. Both variants have a structure similar to DES [15], with a replaced F function and initial and final permutations and a replaced key-scheduling algorithm. The new F function XORs the right half of the data with the subkey and expands the result to 48 bits, which enter into four 12-bit to 8-bit S-boxes. The output of the S-boxes is concatenated and permuted to form the output of the F function. In LOKI89 (see Fig. 1), the initial and the final permutations are replaced by transformations which exclusive-or the data with the key: the initial transformation XORs the plaintext with the key itself and the final transformation XORs the data with the swapped

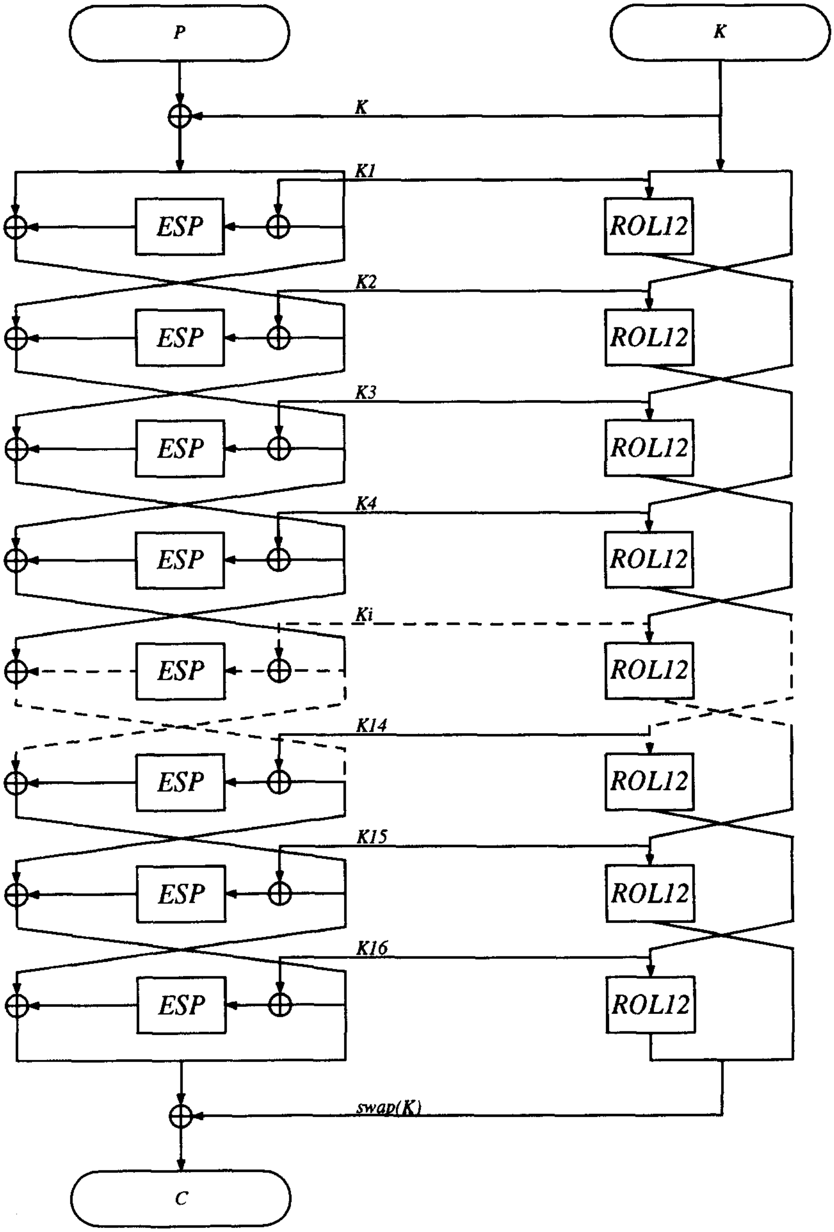


Fig. 1. Outline of LOKI89.

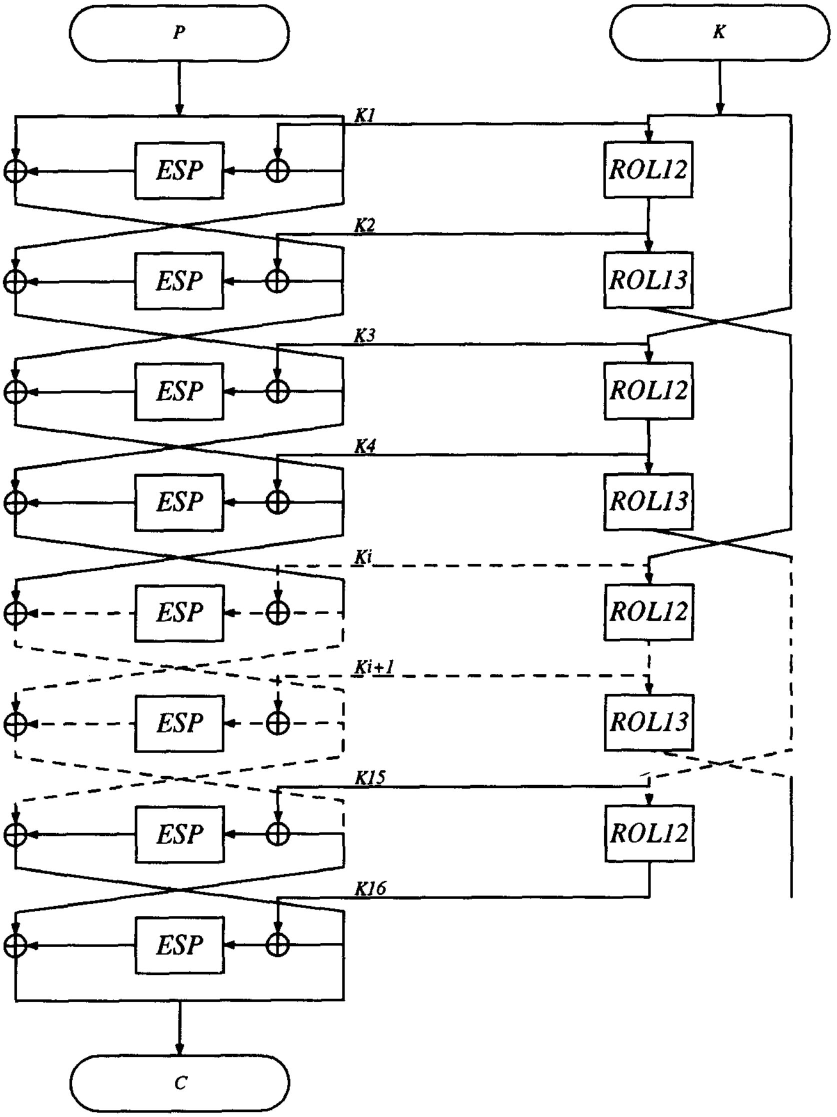


Fig. 2. Outline of LOKI91.

key (whose halves are exchanged). The key-scheduling algorithm takes a 64-bit key, declares its left half as the value of K1 and its right half as the value of K2. Each other subkey Ki (out of K3, ..., K16) is defined by rotating the subkey Kj of round $j = i - 2$ by 12 bits to the left (Ki = ROL12(Kj)). Thus, all the subkeys of the odd rounds share the same bits and all the subkeys of the even rounds share the same bits.

LOKI91 (see Fig. 2) differs from LOKI89 by the choice of the S-boxes, which are chosen to hold better against differential cryptanalysis. The initial and the final permutations are eliminated. The new key-scheduling algorithm declares the value of the left half of the key to be K1 and the same value rotated 12 bits to the left is declared to be K2. The value of the right half of the key is declared to be K3 and the same value rotated 12 bits to the left is declared as K4. Each other subkey Ki (out of K5, ..., K16) is defined by rotating the subkey Kj of round $j = i - 4$ by 25 bits to the left (Ki = ROL25(Kj)). Still, the subkeys share bits with a very structured order.

3. The Chosen Key Attacks

In the chosen key attacks, two related keys with certain relationship are used and several plaintexts are encrypted under each of them. The attacker knows only the relationship between the two keys, but not the keys themselves. He receives the ciphertexts and uses them to find both keys. Two kinds of chosen key attacks are studied: a chosen key known plaintext attack in which only the relation between the keys is chosen by the attacker, and a chosen key chosen plaintext attack in which the attacker chooses the relation between the keys as well as the plaintexts to be encrypted. These attacks are independent of the exact number of rounds of the attacked cryptosystem, and even if the number of rounds is increased (and especially is doubled), the resulting cryptosystem remains vulnerable to the same attack.

3.1. *LOKI89*

In LOKI89 every choice of two subkeys, one from an odd round and one from an even round, have a corresponding 64-bit key. Since all the algorithms for deriving the subkeys from the two preceding subkeys are the same, the position of the rounds in which two subkeys present does not affect the derivation of the following subkeys (nor the preceding ones). If we only fix two subkeys K2 and K3 of a key K, and define a second K* by choosing K1* = K2 and K2* = K3, then the values of the subkeys Ki* of the key K* are the same as the following subkeys K($i + 1$) of the key K. In this case, K* = (K2, K3) = (K$_R$, ROL12(K$_L$)). Therefore, the following property holds for any two such related keys: If the data before the second round in an encryption under the key K equals the data before the first round in an encryption under the key K*, then the data and the inputs of the F functions are the same in both executions shifted by one round. In this case, if the plaintext P is encrypted under the key K, then the data before the second round is $(P_R \oplus \mathrm{K}_R, P_L \oplus \mathrm{K}_L \oplus F(P_R \oplus \mathrm{K}_R, \mathrm{K}_L))$. This data equals the data before the first round in the other encryption under the key K*, whose value is $P^* \oplus \mathrm{K}^* = (P_L^* \oplus \mathrm{K}_R, P_R^* \oplus \mathrm{ROL12}(\mathrm{K}_L))$, and thus in such a pair

$$P^* = (P_R, P_L \oplus \mathrm{K}_L \oplus \mathrm{ROL12}(\mathrm{K}_L) \oplus F(P_R \oplus \mathrm{K}_R, \mathrm{K}_L)). \tag{1}$$

We see that the right half of P equals the left half of P^* and that the relation between the other halves depends on the keys. In such a pair, there is also a similar relation between the ciphertexts:

$$C^* = (C_R \oplus \mathrm{K}_L \oplus \mathrm{ROL12}(\mathrm{K}_L) \oplus F(C_L \oplus \mathrm{K}_R, \mathrm{K}_L), C_L). \tag{2}$$

Figures 3 and 4 describe the relations between the subkeys of the two keys and the relations between the values during the two encryptions.

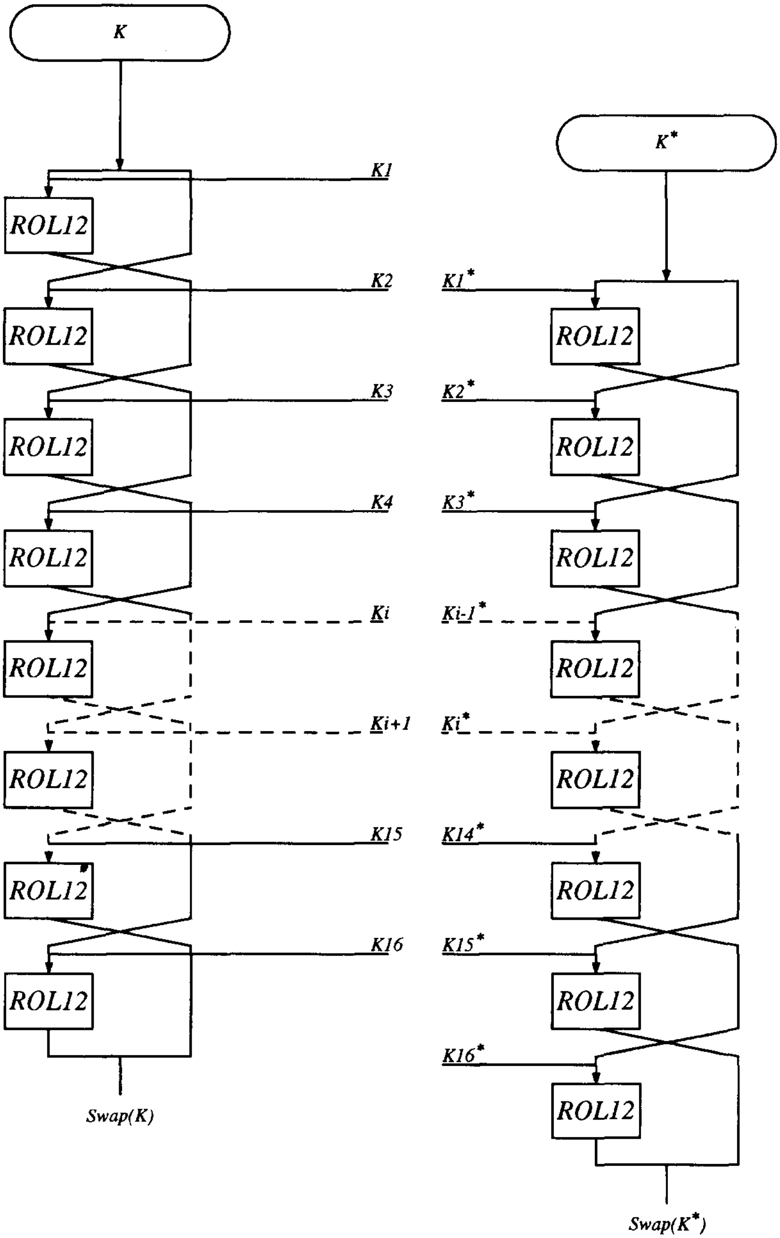


Fig. 3. Relations of subkeys in the key-scheduling algorithm of LOKI89.

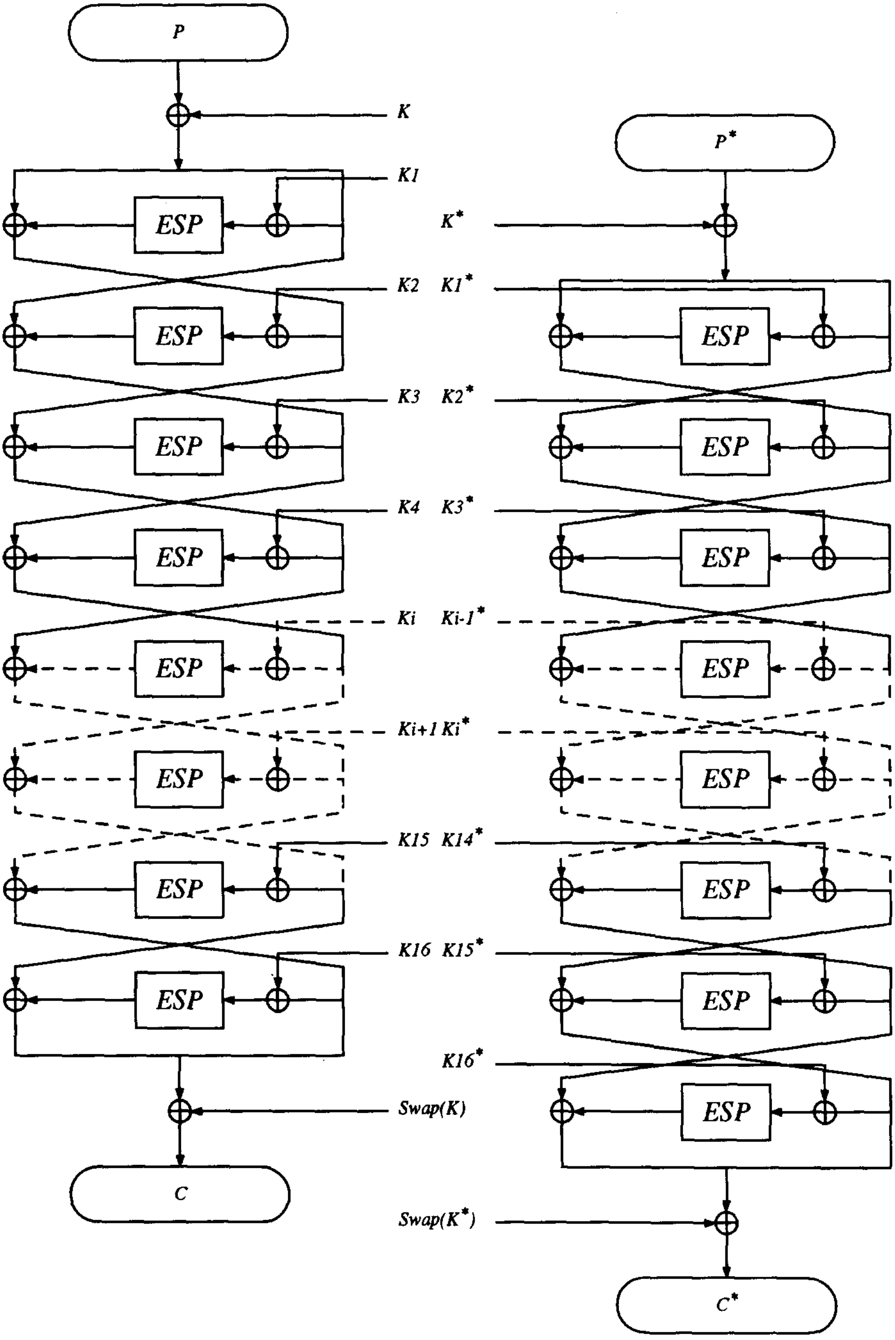


Fig. 4. Relations during LOKI89 encryption.

A chosen key chosen plaintext attack based on this property chooses a 32-bit value P_R, 2^{16} plaintexts $P_0, \ldots, P_{65535}$ whose right halves equal P_R and whose 32-bit left halves are randomly chosen, and 2^{16} plaintexts $P_0^*, \ldots, P_{65535}^*$ whose left halves equal P_R and whose 32-bit right halves are randomly chosen. Two unknown related keys are used to encrypt these plaintexts on the target machine: a key K is used to encrypt the first 2^{16} plaintexts and the key $\mathrm{K}^* = (\mathrm{K}_R, \mathrm{ROL12}(\mathrm{K}_L))$ is used to encrypt the other 2^{16} plaintexts. In every pair of plaintexts P_i and P_j^* we are guaranteed that $P_{jL}^* = P_{iR}$ and by the birthday paradox with a high probability two plaintexts P_i and P_j^* exist such that $P_{jR}^* = P_{iL} \oplus \mathrm{K}_L \oplus \mathrm{ROL12}(\mathrm{K}_L) \oplus F(P_{iR} \oplus \mathrm{K}_R, \mathrm{K}_L)$. In such a pair the data is the same is both executions shifted by one round. It is easy to identify this pair, if it exists, by checking whether $C_R^* = C_L$. This test has a probability of 2^{-32} to pass accidentally, and thus only a few pairs may pass this test.

A pair with this property relates the values of the two plaintexts and of the two ciphertexts to the key by (1) and (2). Thus, such a pair reveals the value of

$$F(P_R \oplus \mathrm{K}_R, \mathrm{K}_L) \oplus F(C_L \oplus \mathrm{K}_R, \mathrm{K}_L) = P_R^* \oplus P_L \oplus C_L^* \oplus C_R$$

in which the only unknown value is $\mathrm{K}_L \oplus \mathrm{K}_R$. Out of the 2^{32} possible values of $\mathrm{K}_L \oplus \mathrm{K}_R$, only few values satisfy the equation. Using differential cryptanalytic [1], [3] optimization techniques, such as difference distribution tables and storing the possible pairs of each of their entries in a special preprocessed table, the identification of the value of $\mathrm{K}_L \oplus \mathrm{K}_R$ can be done in few operations. After we find $\mathrm{K}_L \oplus \mathrm{K}_R$, it is easy to calculate $\mathrm{K}_L \oplus \mathrm{ROL12}(\mathrm{K}_L)$ by (1) and (2) and to derive K and K*.

A similar chosen key known plaintext attack uses 2^{32} known plaintexts P_i encrypted under an unknown key K, and 2^{32} known plaintexts P_j^* encrypted under the related key $\mathrm{K}^* = (\mathrm{K}_R, \mathrm{ROL12}(\mathrm{K}_L))$. By the birthday paradox there is a high probability of having a pair in which the property holds. It is easy to identify this pair by the 32 common bits of the plaintexts and the 32 common bits of the ciphertexts. This pair may be used to reveal the keys in the same way as in the chosen key chosen plaintext attack.

3.2. *LOKI91*

The key-scheduling algorithm of LOKI91 derives the subkeys of the even rounds in a different way than the subkeys of the odd rounds. In particular, the subkey of an even round i is just a rotated value of the subkey of the previous round $i - 1$, and it is independent of the value of round $i + 1$. Thus, we cannot shift the subkeys and the data by one round in the second execution without changing their values. However, we can shift them by two rounds instead. In this case we define the second set of subkeys to be K1* = K3, K2* = K4, and so on. The keys themselves are K and $\mathrm{K}^* = (\mathrm{K}_R, \mathrm{ROL25}(\mathrm{K}_L))$. Since the shift is increased to two rounds, we cannot easily identify the pair we are looking for. Luckily, the two subkeys K1 and K2 share the same 32 bits, and thus, given a plaintext P, we have only to guess 32 bits of the key in order to find the data before the third round, which we define as P^*.

In the chosen key chosen plaintext attack, the attacker chooses a random plaintext P. For each one of the 2^{32} possible values of K1 and K2, he calculates the data before the third round, and defines these 2^{32} values as P_i^*. Given the ciphertexts $C = \text{LOKI91}(P, K)$ and $C_i^* = \text{LOKI91}(P_i^*, K^*)$, he searches for the plaintext P_i^* which was defined by the real values of K1 and K2 by verifying the relationships between the ciphertexts. The subkeys K15* and K16* share the bits of K1 and K2, which he has already guessed. Thus, it is easy to find the real values of K1 and K2 by calculating further from C into C_i^* by each possibility, and only the possibilities which result with the encrypted value of C_i^* might have the real value. After we find these 32 bits of the keys, it is easy to find the other 32 bits by exhaustive search.

The best choice of a chosen key known plaintext attack uses 2^{16} plaintexts P_i and 2^{48} plaintexts P_j^*. With high probability there is pair P_i and P_j^*, for which P_j^* equals the data before the third round during the encryption of P_i. The attacker can identify the key by guessing the 32 bits of K1 and K2 and verifying which of which is possible in a way similar to the chosen key chosen plaintext attack.

4. The Chosen Plaintext Attacks

In this section we describe a chosen plaintext attack which reduces the complexity of exhaustive search using related keys. This attack can be combined with the attacks based on complementation properties [8] and its fastest variant is almost three times faster than the corresponding attacks based only on complementation properties. When this attack is used against 64-bit blockciphers, it requires about 2^{32}–2^{37} chosen plaintexts, whose corresponding ciphertexts are to be stored in random access memory during the analysis.

The idea is similar to the attack based on the complementation property of DES [8]. The complementation property of DES suggests that whenever a plaintext P is encrypted under a key K into a ciphertext $C = \text{DES}(P, K)$, then the complement of P is encrypted by the complement of K into the complement of C: $\overline{C} = \text{DES}(\overline{P}, \overline{K})$. The attack chooses a complementary pair of plaintexts P_1 and $P_2 = \overline{P}_1$. Given their ciphertexts $C_1 = \text{DES}(P_1, K)$ and $C_2 = \text{DES}(P_2, K)$ under the same key K, the attacker searches for the key K by trying all the keys K′ whose most significant bits are zero (i.e., half of the key space). For each such key, he encrypts P_1 into C' by $C' = \text{DES}(P_1, K')$. If $C' = C_1$, it is very likely that K = K′. In addition, the attacker can predict the ciphertext of P_1 under the key $\overline{K}'$ to be $\overline{C}_2$ without an additional encryption. If $C' = \overline{C}_2$, it is very likely that $K = \overline{K}'$, since, due to the complementation property, $\overline{C}_2 = \text{DES}(P_1, \overline{K})$. Otherwise, neither K′ nor $\overline{K}'$ can be the key K. This attack can be carried out even under a known plaintext attack [3], given about 2^{33} known plaintexts, since it is very likely that two complementary plaintexts exist within 2^{33} random plaintexts due to the birthday paradox.

LOKI89 has several complementation properties [2], [3], [4], [11], [9]. A key complementation property causes any key to have 15 equivalent keys which encrypt any plaintext to the same ciphertext. These 15 keys are the original key XORed with the 15 possible 64-bit hexadecimal numbers whose digits are identical. Encryption with these keys results with the same inputs to the F functions in all 16 executions. Therefore, most of the keys are redundant and a known plaintext attack can be carried out with a complexity of 2^{60} rather than 2^{64}.

Another complementation property of LOKI89 is due to the observation that XORing the key with an hexadecimal value $gggggggghhhhhhhh_x$ and XORing the plaintext by $iiiiiiiiiiiiiiii_x$, where $g \in \{0_x, \ldots, F_x\}$, $h \in \{0_x, \ldots, F_x\}$, and $i = g \oplus h$, results in XORing the ciphertext by $iiiiiiiiiiiiiiii_x$. This property can be used to reduce the complexity of a chosen plaintext attack by a further factor of 16 to 2^{56} using the following algorithm:

1. Choose any plaintext P_0, and calculate the 15 plaintexts P_i, $i \in \{1_x, \ldots, F_x\}$, by $P_i = P_0 \oplus iiiiiiiiiiiiiiii_x$.
2. Given the 16 ciphertexts $\{C_i\}$ of the plaintexts P_i under an unknown key K, try all the 2^{56} keys K′ in which eight bits are zero: the four most significant bits of the left half (K'_L) and the four most significant bits of the right half (K'_R).
3. Encrypt the plaintext P_0 by each trial key K′.
4. If the result equals one of the values $C_i \oplus iiiiiiiiiiiiiiii_x$, the original key is likely to be either $K = K' \oplus 00000000iiiiiiii_x$ or any one of its 15 equivalent keys.

The attack we present in this section can predict the values of additional ciphertexts generated from additional plaintexts under related keys. Let P be any plaintext, let K be any key, and let C be the ciphertext $C = \text{LOKI89}(P, \text{K})$. Let $\text{K}^* = (\text{K}_R, \text{ROL12}(\text{K}_L))$ and let P^* be the plaintext whose data before the first round of the encryption under K* is the same as the data before the second round during the original encryption of P under K. Then the first 15 rounds of the encryption $C^* = \text{LOKI89}(P^*, \text{K}^*)$ have exactly the same data and subkeys as the last 15 rounds of the original encryption of P, and the right half of C^* equals the left half of C (i.e., $C_R^* = C_L$).

For each key, there is one equivalent key whose four most significant bits are zero, and one complement key whose four most significant bits of each of its halves are zero. In the following definition the *next* operation rotates a half-key by 12 bits (as is done in the key-scheduling algorithm every round) and finds the supposed equivalent value of the result.

Definition 1. The *next* operation (of LOKI89) takes a 32-bit value, rotates it 12 bits to the left (ROL12) and XORs it with a 32-bit hexadecimal number whose all digits are equal, such that the four most significant bits of the result are zero.

Table 1 shows the cycle size, number of cycles, and the total number of elements

Table 1. Cycles of half-keys for LOKI89.

Cycle size	Number of cycles	Number of elements in the cycles
1	16	16
2	120	240
4	16,320	65,280
8	33,546,240	268,369,920
Total	33,562,696	268,435,456*

*268,435,456 = 2^{28}.

in the cycles generated by the *next* operation (of LOKI89). We see that almost all the cycles have the maximal size eight.

For simplicity and clarity, we start showing a variant of the related keys attack which halves the complexity of a chosen plaintext attack into 2^{55}. For this attack we preprocess a list of about 2^{27} 32-bit potential half-keys $\{L_i\}$, with the properties:

1. The four most significant bits of all the values in the list are zero.
2. The list contains at least one value from any pair L_i and L_j, for which $L_i = \text{next}(L_j)$ (i.e., $L_i = \text{ROL12}(L_j) \oplus hhhhhhhh_x$ for some hexadecimal digit h).
3. The list is minimal according to the above properties.

In Fig. 5 we show a program that creates such a list with $2^{27} + 8$ values. A similar program was used to verify Table 1 by counting the number of cycles, rather than inserting to the list, and was run in about an hour on a personal computer. The list

```
#define ROL(v, b) (((v)<<(b)) | ((v)>>(32-(b))))
#define next(i) (ROL(i, 12) ^ lmask[(i>>16) & 0xF])

unsigned long lmask[16] =
    {0x00000000L, 0x11111111L, 0x22222222L, 0x33333333L,
     0x44444444L, 0x55555555L, 0x66666666L, 0x77777777L,
     0x88888888L, 0x99999999L, 0xAAAAAAAAL, 0xBBBBBBBBL,
     0xCCCCCCCCL, 0xDDDDDDDDL, 0xEEEEEEEEL, 0xFFFFFFFFL };

main()
{
  unsigned long i1;

  for(i1=0; i1<(1L<<28); i1++)
    {
      unsigned long i2 = next(i1);
      unsigned long i3 = next(i2);
      unsigned long i4 = next(i3);
      unsigned long i5 = next(i4);
      unsigned long i6 = next(i5);
      unsigned long i7 = next(i6);
      unsigned long i8 = next(i7);
      if(i1 == min(i1, i2, i3, i4, i5, i6, i7, i8))
        {
          Insert i1 to the list.
          if(i1 != i3) Insert i3 to the list.
          if(i1 != i5) Insert i5 and i7 to the list.
        }
    }
}
```

Fig. 5. A program to create the list of potential half-keys.

can either be stored in memory using about $4 \cdot 2^{27} = 2^{29}$ bytes, or stored as a bitmap using $2^{28}/8 = 2^{25}$ bytes.[1] The attack itself requires 2^{36} chosen plaintexts which should be stored in a random access memory (whose size is 2^{39} bytes). The attack is as follows:

1. Choose any plaintext P_0.
2. Calculate the 15 plaintexts P_i, $i \in \{1_x, \ldots, F_x\}$, by $P_i = P_0 \oplus iiiiiiiiiiiiiiii_x$.
3. For each plaintext P_i, choose the additional 2^{32} plaintexts $P_{i,k} = (P_{iR}, P_{iL} \oplus k)$ whose left halves are the right half of P_i and whose right halves receive all the possible values by XORing all the possible 32-bit values k to the left half of P_i.
4. Given the ciphertexts $\{C_i\}$, $\{C_{i,k}\}$, try all the 2^{55} keys K′ of the forms: K′ = (L_i, L_j) and K′ = (ROL12(L_i), ROR12(L_j)).
5. Encrypt the plaintext P_0 under each trial key K′ into C' = LOKI89(P_0, K′).
6. If C' equals one of the values $C_i \oplus iiiiiiiiiiiiiiii_x$, the original key is likely to be either K = K′ $\oplus$ 00000000$iiiiiiii_x$ or any one of its 15 equivalent keys.
7. The output of the F function in the first round of the encryption of P_0 under K′ equals exactly one k. If C'_L equals one of the values $C_{i,kR} \oplus iiiiiiii_x$, continue encryption of P_0 with a 17th round (just calculate one additional round from C' using the subkey K17′ which can be easily derived from the key K′), and if the result C'' equals $C_{i,k} \oplus iiiiiiiiiiiiiiii_x$, then the original key is likely to be K = (K′$_R$, ROL12(K′$_L$) $\oplus$ $iiiiiiii_x$) or any one of its 15 equivalent keys.

The encryption key K must be recognized by either step 6 or step 7. Since the comparisons in steps 6 and 7 are much faster than a trial encryption, the complexity of the attack is the total complexity of step 5, which is 2^{55}.

The complexity of this attack can be reduced further using the observation that the shift of one round of the data and the subkeys can be done both in the backward and forward directions simultaneously, rather than in one direction only. Therefore, each trial key can predict ciphertexts under three related keys, and thus the attack requires trying about a third of the number of keys required by the attack based on the complementation properties. Usually it is not possible to find a subset of the keys which satisfy the related keys conditions and contain exactly a third of all the keys. The best choice for LOKI89 is to try three-eighths of the keys. We preprocess a list of about 2^{25} half-keys $\{L_i\}$, with the properties:

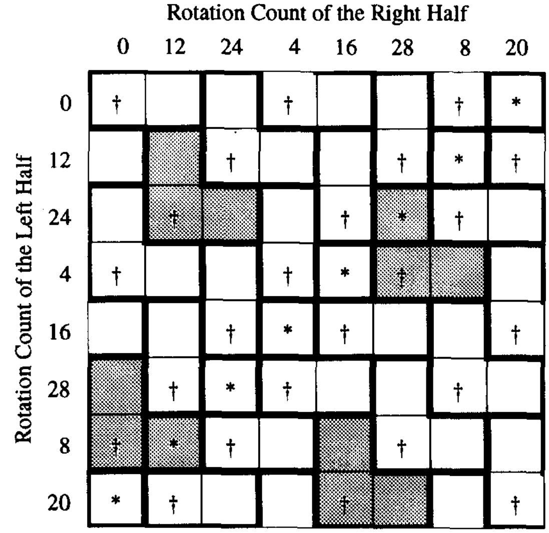
1. The four most significant bits of all the values in the list are zero.
2. The list contains exactly one value from each cycle of the next operation.

The program which creates the list for this attack is very similar to the program described in Fig. 5. The difference between them is only that instead of inserting several values from each cycle, only one value (il) is inserted to the list by this program.

[1] We have also devised an additional algorithm which chooses the trial keys on the fly and does not require a list of potential half-keys. However, due to technical details, the efficiency of the attack is reduced. Since the memory space required by the attack is much larger than the list of potential half-keys, this list does not affect the space complexity of the attack.

The attack itself requires 2^{37} chosen plaintexts which should be stored in a random access memory (whose size is 2^{40} bytes). The attack is as follows:

1. Choose any plaintext P_0, and calculate the 15 plaintexts P_i, $i \in \{1_x, \ldots, F_x\}$, by $P_i = P_0 \oplus iiiiiiiiiiiiiiii_x$.
2. For each plaintext P_i, choose the additional 2^{32} plaintexts $P_{i,k} = (P_{iR}, P_{iL} \oplus k)$ whose left halves are the right half of P_i and whose right halves receive all the possible values by XORing all the possible 32-bit values k to the left half of P_i.
3. For each plaintext P_i, choose the additional 2^{32} plaintexts $P^*_{i,k} = (P_{iR} \oplus k, P_{iL})$ whose right halves are the left half of P_i and whose left halves receive all the possible values by XORing all the possible 32-bit values k to the right half of P_i.
4. Given the ciphertexts $\{C_i\}$, $\{C_{i,k}\}$, $\{C^*_{i,k}\}$, try for each pair of half-keys (L_i, L_j) all the 24 keys K′ of the form K′ = (ROR$m(L_i)$, ROL$n(L_j)$), where m is a multiple of four and n is either $n = m$, $n = m + 4$, or $n = m + 8$. Figure 6 shows the choice of such 24 trial keys, which are denoted by †. All the keys are covered by the trial keys. The trial keys K′ are covered by themselves. The other keys are covered by the key relation property by trial keys created from (L_j, L_i). These keys are surrounded together with the swapped value of their trial keys. Examples of four such triples are marked in gray. The keys denoted by * are covered by two trial keys. The figure should be interpreted cyclically through its edges. This is the best coverage possible for LOKI89.
5. Encrypt the plaintext P_0 under each trial key K′ into $C' = \text{LOKI89}(P_0, \text{K}')$.
6. If C' equals one of the values $C_i \oplus iiiiiiiiiiiiiiii_x$, the original key is likely to be either K = K′ ⊕ $00000000iiiiiiii_x$ or any one of its 15 equivalent keys.
7. Fix k to be the output of the F function in the first round of the encryption of P_0 under the key K′. If C'_L equals one of the 16 values $C_{i,kR} \oplus iiiiiiii_x$, continue encryption of P_0 with a 17th round (just calculate one additional round from



Rotation Count of the Right Half

Rotation Count of the Left Half	0	12	24	4	16	28	8	20
0	†			†			†	*
12		▨	†			†	*	†
24		▨ †	▨		†	▨ *	†	
4	†			†	*	▨ †	▨	
16			†	*	†			†
28	▨	†	*	†			†	
8	▨ †	▨ *	†		▨	†		
20	*	†			▨ †	▨		†

Fig. 6. The choice of the trial keys. †, The 24 trial keys; *, keys covered by two trial keys.

C' using the subkey K17$'$ which can be easily derived from the key K$'$), and if the result C'' equals $C_{i,k} \oplus iiiiiiiiiiiiiiii_x$, then the original key is likely to be $\mathrm{K} = (\mathrm{K}'_R, \mathrm{ROL12}(\mathrm{K}'_L) \oplus iiiiiiii_x)$ or any one of its 15 equivalent keys.

8. Calculate one additional round backward from P_0 using the subkey K0$'$ which can be easily derived from the key K$'$, and fix k to be the output of the F function in this round. If the data after the 15th round during the encryption of P_0 under the key K$'$ equals one of the 16 values $C_{i,k} \oplus iiiiiiiiiiiiiiii_x$ (for some i), the original key is likely to be $\mathrm{K} = (\mathrm{ROR12}(\mathrm{K}'_R), \mathrm{K}'_L \oplus iiiiiiii_x)$ or any one of its 15 equivalent keys.[2]

The encryption key K must be recognized by either steps 6, 7, or 8. Since the comparisons in steps 6, 7, and 8 are much faster than trial encryptions, the complexity of the attack is the total complexity of step 5. If the number of elements in each cycle would have been divisible by three, the complexity of the attack would decrease relatively to the attack based on complementation properties by a factor of $\frac{1}{3}$. In the case of LOKI89 this factor is only $\frac{3}{8} = 0.375$, and the complexity of the attack is $\frac{3}{8} \cdot (2^{28})^2 = 1.5 \cdot 2^{54}$ trial encryptions.[3]

The application of this attack to LOKI91 is similar. The two main differences are:

(1) The plaintexts are chosen with respect to changes in the half of the key which affects the first two rounds, rather than changing the plaintext's halves directly (since we must change the halves of the plaintexts in this case by predicting the data after two rounds).

(2) The trial keys generated in step 4 of the attack cover only half of the keys, since there is only one complementation property. Thus, we use twice as many trial keys, namely the original trial keys and the same trial keys whose right halves are complemented.

Table 2 shows the number of cycles with each cycle size and the total number of elements in the cycles of the *next* operation of LOKI91. Almost all the cycles have the maximal size 32. The complexity of this attack on LOKI91 is $1.375 \cdot 2^{61}$, which is about 5.8 times faster than exhaustive search.

5. Application to DES

DES [15] is not vulnerable to the related keys attacks since the number of shifts in the key-scheduling algorithm is not the same in all the rounds. While the key registers are usually shifted by two bits after each round, they are shifted only by one bit after the first, the eighth, and the fifteenth rounds.[4] However, if we modify this shift pattern to shift the key registers by the same number of bits in all the rounds

[2] This part of the algorithm requires the calculation of one additional round for any trial key K$'$, which slows the attack by a factor of $\frac{17}{16}$. In parallel hardward implementations this behavior can be solved easily. In software it can also be solved by using larger hash tables and checking the existence of ciphertexts before the calculation of the additional round.

[3] If we carefully choose the order of trying the trial keys, we can reduce the average case complexity slightly from $1.5 \cdot 2^{53}$ by about 8%.

[4] Actually, DES has related keys with shifts of seven rounds, which cannot be used to design an attack.

Table 2. Cycles of half-keys for LOKI91.

Cycle size	Number of cycles	Number of elements in the cycles
1	2	2
2	1	2
4	3	12
8	30	240
16	4,080	65,280
32	67,106,816	2,147,418,112
Total	67,110,932	2,147,483,648*

* 2,147,483,648 = 2^{31}.

(either one or two or any other number including seven which was suggested in [6]), the resultant cryptosystem becomes vulnerable to the related keys attacks. Table 3 describes the number of shifts before each round in the key-scheduling algorithm of DES and a variant with modified shift numbers which is vulnerable to the related keys attacks.

6. Applications to Other Ciphers

The applicability of the related keys attacks to additional cryptosystems suggested in the cryptographic literature are studied in this section. Many such cryptosystems are vulnerable to these attacks, while several others are not. Examples of several such cryptosystems are given below. The details of these vulnerabilities and the invulnerabilities may suggest some ideas on how to immunize against these attacks.

In Lucifer [17] the subkeys are derived by shifting the key by a fixed number of bits every round and selecting specific bits as the subkeys, and thus the derivation algorithm of the subkeys from the previous subkeys is the same in all the rounds. Therefore, chosen key attacks similar to the attacks on LOKI89 are applicable, but since the blocksize and the size of the key are 128 bits, the complexities become 2^{33} for the chosen key chosen plaintext attack and 2^{65} for the chosen key known plaintext attack. Lucifer has no complementation properties, and thus the complexity of the chosen plaintext attack is $1.5 \cdot 2^{126}$.

The other variant of Lucifer [7] has no specific definition of the key-scheduling algorithm (as well as several other parameters of the cipher). If the key-scheduling algorithm is similar to the one of the preceding variant and the blocksize is also 128

Table 3. The numbers of shifts in the key-scheduling algorithm of DES, and a modified variant vulnerable to the related keys attacks.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Modified variant	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

bits, a chosen key chosen plaintext attack similar to the one on LOKI91 (but which shifts only one round) which finds the first (32-bit) subkey is applicable with complexity 2^{32}. Since the subkeys are very short relative to the key, we can relate keys which shift two rounds backward and forward, in addition to the one-round shifts. We can also choose four sets of 2^{32} chosen plaintexts and find 128 key bits with a total complexity of 2^{34}. A chosen key known plaintext attack is applicable with complexity 2^{66}. The complexity of the chosen plaintext attack can be even smaller than $1.5 \cdot 2^{126}$ since more than two additional keys can be verified by each trial encryption (again, since the subkeys are short).

The IDEA cipher [12], [13] is one of the more promising ciphers today. The key-scheduling algorithm of IDEA has key relation properties, but these relations must shift the subkeys by multiples of four rounds, and thus make the attacks impractical. In addition, several other properties of IDEA prohibit the related keys attacks:

- Since IDEA does not swap the halves of the data, the complexity of a chosen key chosen plaintext attack would be at least $2^{k/2}$ (which for IDEA is 2^{64}), rather than $2^{k/4}$.
- IDEA does not allow easy identification of the related pairs. In DES-like ciphers the identification can be based on the swapping structure and the fact that one half in one ciphertext (or a simple function of it) has the same value as the other half in the other ciphertext (or the same simple function of it).
- The round operations are used in a more complex way (the output of one operation becomes the input to another operation in the same round) and thus the analysis and identification of the pairs become less effective.

FEAL [16], [14] is not vulnerable to the related keys attacks although its key processing uses the same algorithms to derive the subkeys from the previous subkeys, since most lists of related subkeys are not derivable from some key (due to the initialization of the key processing), and since the initial and the final transformations use subkeys calculated in a complex way from the key (so the identification of the related pairs becomes intractable).

7. Summary

In this paper we described new cryptanalytic attacks which are applicable to the LOKI family of blockciphers and to Lucifer. These new attacks are based on the structure of the key-scheduling algorithms. Since we assume that in all the intermediate rounds the data and the subkeys are the same in both executions, with a difference of one (or two) rounds, this attack is independent of the number of the rounds of the cipher. The same attacks could be applicable to DES if only minor changes were made to the shift pattern of its key-scheduling algorithm, and thus these attacks show how šo small points in the design of a cipher can contribute to its strength. The results of the related keys attacks are summarized in Table 4.

Table 4. Results of the related keys attacks.

	LOKI89	LOKI91	Lucifer	Modified DES*
Chosen plaintext: related keys				
Complexity of attack	$1.5 \cdot 2^{54}$	$1.375 \cdot 2^{61}$	$1.5 \cdot 2^{126}$	$1.43 \cdot 2^{53}$
Chosen plaintexts required	2^{37}	2^{34}	2^{65}	2^{34}
Chosen key				
Chosen plaintexts	2^{17}	2^{32}	2^{33}	2^{17}
Known plaintexts	2^{33}	2^{48}	2^{65}	2^{33}
Previously known attacks				
Differential cryptanalysis	—	—	—	2^{47}
Complementation properties†	2^{56}	2^{63}	—	2^{55}
Exhaustive search‡	2^{60}	2^{64}	2^{128}	2^{56}

* The shift pattern of the key registers in the key-scheduling algorithm is modified to have the same number of shifts in all the rounds (see Table 3).

† Either two chosen plaintexts are required or 2^{33} known plaintexts. For LOKI89, 16 chosen plaintexts are required to achieve this complexity.

‡ One known plaintext is required.

Open Problem. The 50% reduction of the complexity of exhaustive search of DES by a chosen plaintext attack using the complementation property [8] can be converted into a known plaintext attack [3] with the same complexity. It is an open problem whether the chosen plaintext attack described in this paper can be converted into a known plaintext attack.

References

[1] E. Biham and A. Shamir, Differential Cryptanalysis of DES-like Cryptosystems, *Journal of Cryptology*, Vol. 4, No. 1, pp. 3–72, 1991.

[2] E. Biham and A. Shamir, Differential Cryptanalysis of Snefru, Khafre, REDOCII, LOKI and Lucifer, Technical Report CS91-18, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, 1991. The extended abstract appears in *Advances in Cryptology, Proceedings of CRYPTO '91*, pp. 156–171, Lecture Notes in Computer Science, Vol. 576, Springer-Verlag, Berlin, 1992.

[3] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, New York, 1993.

[4] L. Brown, M. Kwan, J. Pieprzyk, and J. Seberry, Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI, *Advances in Cryptology, Proceedings of ASIACRYPT '91*, pp. 36–50, Lecture Notes in Computer Science, Vol. 739, Springer-Verlag, Berlin, 1993.

[5] L. Brown, J. Pieprzyk, and J. Seberry, LOKI—A Cryptographic Primitive for Authentication and Secrecy Applications, *Advances in Cryptology, Proceedings of AUSCRYPT '90*, pp. 229–236, Lecture Notes in Computer Science, Vol. 453, Springer-Verlag, Berlin, 1990.

[6] L. Brown and J. Seberry, Key Scheduling in DES-Type Cryptosystems, *Advances in Cryptology, Proceedings of AUSCRYPT '90*, pp. 221–228, Lecture Notes in Computer Science, Vol. 453, Springer-Verlag, Berlin, 1990.

[7] H. Feistel, Cryptography and Data Security, *Scientific American*, Vol. 228, No. 5, pp. 15–23, May 1973.

[8] M. E. Hellman, R. Merkle, R. Schroppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer,

Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard, Technical Report, SEL 76-042, Stanford University, September 1976.

[9] L. R. Knudsen, Cryptanalysis of LOKI, *Advances in Cryptology, Proceedings of ASIACRYPT '91*, pp. 22–35, Lecture Notes in Computer Science, Vol. 739, Springer-Verlag, Berlin, 1993.

[10] L. R. Knudsen, Cryptanalysis of LOKI91, *Advances in Cryptology, Proceedings of AUSCRYPT '92*, pp. 196–208, Lecture Notes in Computer Science, Vol. 718, Springer-Verlag, Berlin, 1993.

[11] M. Kwan and J. Pieprzyk, A General Purpose Technique for Locating Key Scheduling Weakness in DES-Like Cryptosystems, *Advances in Cryptology, Proceedings of ASIACRYPT '91*, pp. 237–246, Lecture Notes in Computer Science, Vol. 739, Springer-Verlag, Berlin, 1993.

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

[13] X. Lai, On the Design and Security of Block Ciphers, Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich, 1992.

[14] S. Miyaguchi, A. Shiraishi, and A. Shimizu, Fast Data Encryption Algorithm FEAL-8, *Review of Electrical Communications Laboratories*, Vol. 36, No. 4, pp. 433–437, 1988.

[15] National Bureau of Standards, Data Encryption Standard, FIPS Publication 46, U.S. Department of Commerce, January 1977.

[16] A. Shimizu and S. Miyaguchi, Fast Data Encryption Algorithm FEAL, *Advances in Cryptology, Proceedings of EUROCRYPT '87*, pp. 267–278, Lecture Notes in Computer Science, Vol. 304, Springer-Verlag, Berlin, 1987.

[17] A. Sorkin, Lucifer, a Cryptographic Algorithm, *Cryptologia*, Vol. 8, No. 1, pp. 22–41, January 1984.