On Weaknesses of Non–surjective Round Functions

Vincent Rijmen* Bart Preneel†
Katholieke Universiteit Leuven, ESAT-COSIC
K. Mercierlaan 94, B-3001 Heverlee, Belgium

{vincent.rijmen,bart.preneel}@esat.kuleuven.ac.be

Extended Abstract

Abstract

We propose a new attack on Feistel ciphers with a non-surjective round function. CAST and LOKI91 are examples of such ciphers. We extend the attack towards ciphers that use a non–uniformly distributed round function and apply the attack to CAST.

1 Introduction

The Feistel structure is a very common structure for block ciphers, the most prominent example being the Data Encryption Standard [FI46]. Although DES has been a worldwide de facto standard since 1977, everybody agrees that it is reaching the end of its life time. The main reason is the size of the key, which is only 56 bits. The key size was already a topic of discussion in the seventies [DH77], and it was shown recently by M. Wiener that at present an exhaustive key search in 3.5 hours requires only 1 million US$ of equipment [W93]. Of more theoretical interest are recent cryptanalytic techniques such as differential [BS93] and linear [Ma93a, Ma94] cryptanalysis which provide techniques to recover the key faster than exhaustive search. Currently, they do not offer a threat for practical applications, but it can be expected that

*N.F.W.O. research assistant, sponsored by the National Fund for Scientific Research (Belgium).

†N.F.W.O. postdoctoral researcher, sponsored by the National Fund for Scientific Research (Belgium).

within the next five years practical attacks are developed. These problems can be overcome easily by using triple DES with two keys, at the cost of a reduced performance.

A second problem of the DES is the fact that it was designed taking into account 1977 hardware constraints. In spite of this, very fast software implementations have been reported (7 Mbit/s on a 80586/60MHz and 12 Mbit/s on a HP-715/80). However, algorithm designers hope to exploit in a more efficient way the present day computer architectures, and to achieve a better tradeoff between security and speed. In order to build on the experience gathered with the cryptanalysis of DES, most designers prefer to keep the Feistel structure. Examples of such proposals are FEAL [M91], LOKI91 [LOKI91], Blowfish [S94], and CAST [AT93, HT94, A94]. By introducing new structures for the round function, designers try to improve the performance and to reduce the vulnerability to differential and linear attacks. However, this might introduce new vulnerabilities, especially if the number of rounds is reduced in order to optimize the speed.

In this extended abstract we will concentrate on the weaknesses that are introduced by the use of non–surjective or, more general, non–uniform round functions. Several studies revealed that in general large S-boxes are more resistant against linear or differential cryptanalysis. It is even argued that one can choose random S-boxes and obtain a secure cipher. We show that this is not always true. In section 2 we describe the general principle of our attack. In section 3 we apply the attack to CAST and LOKI91. In section 4 we conclude by discussing some design principles.

2 General principle

We first explain our notation and then we present the attack and an extension.

2.1 Notation

Consider a Feistel cipher, consisting of n rounds (with n even). The plaintext input consists of two p–bit blocks L_0 and R_0, the key is denoted by K, the ciphertext by (L_n, R_n). Each round takes a $2p$–bit message input block (L_i, R_i) and a k–bit key input (K_i). The round output is given by:

$$\begin{aligned} R_i &= L_{i-1} \oplus F_i(K_i \oplus R_{i-1}) \\ L_i &= R_{i-1} \qquad\qquad\qquad\qquad i = 1, \ldots, n-1. \end{aligned}$$

For the last round (no swapping) this becomes:

$$\begin{aligned} L_n &= L_{n-1} \oplus F_n(K_n \oplus R_{n-1}) \\ R_n &= R_{n-1}\,. \end{aligned}$$

Then the following relation holds:

$$\beta_n(L_0, R_0, K) = \bigoplus_{i=1}^{n/2} F_{2i}(K_{2i} \oplus R_{2i-1}) = R_0 \oplus L_n \,. \tag{1}$$

For unbalanced round functions F_{2i}, the sum β_n will be unbalanced if we assume that the round keys are independent. We expect that this also holds for most key schedulings. Since not all values of β_n have the same probability, an attacker gathers statistical information about the plaintext by looking at the ciphertext.

2.2 Basic attack

If we take the last round out of the sum, (1) becomes

$$\beta_{n-1}(L_0, R_0, K) = \bigoplus_{i=1}^{n/2-2} F_{2i}(K_{2i} \oplus R_{2i-1}) = R_0 \oplus L_n \oplus F_n(K_n \oplus R_n) \,. \tag{2}$$

Non–surjective round functions F_{2i} will result in a non–surjective β_{n-2} for small enough values of n. This is quantified in the following lemma.

Lemma 1 *Denote by f the fraction of p-bit vectors that are a possible output of the round function, and by f_{n-2} the fraction of possible values for β_{n-2}. If the round functions are independent:*

$$f_{n-2} = 1 - (1 - f_{n-4} \cdot f)^{2^p} \,. \tag{3}$$

Proof: We can write

$$\beta_{n-2} = \beta_{n-4} \oplus F_{n-2} \,.$$

A value X is a possible value for β_{n-2} iff

$$X = Y \oplus Z \,, \tag{4}$$

and Y, Z are possible values for β_{n-4} and F_{n-2} respectively. There are 2^p solutions for (4). A value for β_{n-2} is impossible iff for all solutions (X, Y) holds that X or Y is impossible. By application of the product rule we obtain

$$1 - f_{n-2} = (1 - f_{n-4} \cdot f)^{2^p} \,.$$

■

A non–surjective β_{n-2} makes the following attack possible. For all values K_n calculate the right hand side of (2) by use of the known plaintext R_0 and the ciphertext L_n. Check whether this is a possible value for β_{n-2}. Wrong key guesses will eventually produce a value that is outside the range of β_{n-2}.

Since there are 2^k possible round keys K_n, we need on average $-k/\log_2(f_{n-2})$ pairs to determine the right value of K_n. The work factor of the attack is $2^k/(1-f_{n-2})$.

For small values of k, one can search for several round keys at once. This way, f_{n-4} can be used instead of f_{n-2}.

2.3 Statistical attack

Equation (3) shows that for larger values of n, f_{n-2} goes very fast to 1. But β_{n-2} will not be uniformly distributed: all outputs are possible, but they don't occur with the same probability. For still larger values of n, β_{n-2} becomes close to a "random function", which should be a design goal. Our attack can be modified to deal with surjective but unbalanced β_n's. First calculate the relative probabilities for each possible value of β_{n-2}. Then calculate the right hand side of (2) for every value of K_n and for every known plaintext–ciphertext pair. It is now possible to calculate the a posteriori probability for the key candidates.

By Bayes' rule we can express the probability $\Pr(K_n|R_0, L_n)$ that K_n is the correct key, given R_0 and L_n:

$$\Pr(K_n|R_0, L_n) = \frac{\Pr(K_n)\Pr(R_0, L_n|K_n)}{\Pr(R_0, L_n)} = \frac{\Pr(K_n)\Pr(\beta_{n-2})}{\Pr(\beta_n)}.$$

Let us denote with $\Pr^i(K_n)$ the probability that K_n is the right key after the processing of the i–th known plaintext ($\Pr^0(K_n) = 1/2^k$). We have

$$\Pr^i(K_n) = \frac{\Pr^{i-1}(K_n)\Pr(\beta^i_{n-2})}{\Pr(\beta^i_n)} = \frac{1}{2^k}\prod_{j=1}^{i}\frac{\Pr(\beta^j_{n-2})}{\Pr(\beta^j_n)}.$$

This expression can be evaluated for each key candidate and assigns to each round key a probability that can be used for a ranking of the most probable keys.

3 Application to CAST and LOKI91

3.1 CAST

The round function of CAST is constructed as follows: if $b_1b_2b_3b_4$ denotes the four byte input, the output is obtained by adding the output of the four S-boxes:

$$F(b_1b_2b_3b_4) = S_1[b_1] \oplus S_2[b_2] \oplus S_3[b_3] \oplus S_4[b_4].$$

The four S_i are tables with eight input and 32 output bits. Since each S-box has only eight input bits, its output can only take 256 values in GF(2^{32}). If the four S-boxes are selected at random, the expected number of possible

outputs is $e^{-1} \times 2^{32}$, where e denotes the natural logarithm base. This value can also be computed from (3), since adding the outputs of the S-boxes corresponds to concatenating rounds. Table 1 gives the f-values for the combination of 1, 2, 3, and 4 S-boxes.

# S-boxes	f
1	5.96×10^{-8}
2	1.53×10^{-5}
3	3.90×10^{-3}
4	6.32×10^{-1}

Table 1: f-values for the combination of 1 to 4 S-boxes.

The CAST S-boxes are constructed from eight–bit bent functions that are the Walsh transforms of the concatenation of four six–bit bent functions. We constructed S-boxes following this design principle and obtained the same value for f.

We can summarize the CAST key scheduling in the following way: for each round first an "initial value" of two bytes is calculated from the master key. This calculation is simple for the first rounds, and more complicated for the last round. These two bytes are expanded in a non-linear way to the 32–bit round key. The entropy of each round key is therefore at most 16 bits. This enables us to search for three round keys at once.

We can apply the simple attack on six rounds of CAST. Equation (2) becomes:

$$\beta_4 = F_2 = R_0 \oplus L_6 \oplus F(K_4 \oplus R_6 \oplus F(K_5 \oplus L_6 \oplus F(K_6 \oplus R_6))) \oplus F(K_6 \oplus R_6). \quad (5)$$

R_0 is a part of the plaintext, L_6 and R_6 form the ciphertext. K_4, K_5, and K_6 are the round keys we are searching for. Note that by swapping plaintext and ciphertext, we can apply the same attack to find K_1, K_2, and K_3. The work factor of the attack is then 1.5×2^{48}. The number of required texts is only $-\log(2^{48})/\log(1 - e^{-1}) \approx 82$. Note that in [HT94] it is estimated that the required number of known plaintexts to break six rounds of CAST with a linear attack is at least 2^{18}.

Since the sum of two CAST round functions is surjective, the simple attack is not applicable to more than six rounds. The statistical attack needs a table of size 2^{32}. Although this is not infeasible, we are currently unable to actually implement this attack. We are developing an implementation for a mini-version of CAST that operates on a four byte input and with S-boxes that consist of 16 4–bit functions.

3.2 LOKI91

The round function of LOKI91 takes a 32–bit message input and exors this with a 32–bit round key. These 32 bits are expanded to 48 bits and split into four parts. Each part enters the 12×8–bit S-box. This produces the $8 \times 4 = 32$ output bits. Note that of the 48 input bits to the nonlinear part, 32 bits are pairwise equal. In [Kn94] L. R. Knudsen observed that this implies that the output can only take a fraction of $\frac{8}{13}$ of the possible values.

Each round key consists of 32 bits. The key scheduling of LOKI91 is such that $K_{2i} = K_{2i-1} <<< 12$, $i = 1, 2, \ldots, 8$, where $<<<$ denotes "left wise rotation." Therefore we can search for the round keys of two rounds at once, and apply the basic attack to five rounds of LOKI91. We did not implement the statistical attack for LOKI91. Since f is about the same for LOKI91 and CAST, we expect comparable results, except for the fact that we only can peel off two rounds.

4 Discussion

We have shown that the use of uniformly distributed round functions is probably a good design criterion for Feistel ciphers. Feistel ciphers that make use of non–surjective round functions should use a number n of rounds that is large enough to make β_{n-2} at least surjective. In order to counter the statistical attack, the sum should have a distribution which is close to uniform. We conjecture that the deviations of the different outputs squared approximates the number of required known plaintexts. Therefore this type of attack will become infeasible for a large number of rounds.

With respect to the key scheduling of CAST [A94], we can say that round keys with 16 bit entropy are inadequate. The computational cost for an attacker to peel off several rounds is too low. This makes CAST more vulnerable to our attack than LOKI91.

5 Acknowledgement

We wish to thank L.R. Knudsen for helpful comments on the application of the attack to LOKI91.

References

[AT93] C.M. Adams and S.E. Tavares, "Designing S-boxes for ciphers resistant to differential cryptanalysis," *Proc. of the 3rd symposium on State and Progress of Research in Cryptography*, W. Wolfowicz, Ed., Fondazione Ugo Bordoni, 1993, pp. 181–190.

[A94] C.M. Adams, "Simple and effective key scheduling for symmetric ciphers," *Proc. of SAC'94, Workshop on Selected Areas in Cryptography.*

[BS93] E. Biham and A. Shamir, *"Differential Cryptanalysis of the Data Encryption Standard,"* Springer-Verlag, 1993.

[DH77] W. Diffie and M. Hellman, "Exhaustive cryptanalysis of the NBS data encryption standard," *Computer*, pp. 74–78, 1977.

[DM95] D. Davies and S. Murphy, "Pairs and triples of DES S-boxes," *Journal of Cryptology*, Vol. 8, No. 1, 1995, pp. 1–26.

[FI46] *"Data Encryption Standard,"* Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[HT94] H.M. Heys and S.E. Tavares, "On the security of the CAST encryption algorithm," *Canadian Conference on Electrical and Computer Engineering*, Sept. 1994, Halifax, Canada.

[Kn94] L.R. Knudsen, "Block ciphers – analysis, design and applications," *PhD. Thesis, DAIMI PB–485*, Aarhus University, 1994.

[LOKI91] L. Brown, M. Kwan, J. Pieprzyk and J. Seberry, " Improving resistance against differential cryptanalysis and the redesign of LOKI," *Advances in Cryptology, Proc. AsiaCrypt'91, LNCS 453*, H. Imai, R. L. Rivest and T. Matsumoto, Eds., Springer-Verlag, 1993, pp. 36–50.

[Ma93a] M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology, Proc. Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 386–397.

[Ma94] M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard," *Advances in Cryptology, Proc. Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 1–11.

[M91] S. Miyaguchi, "The Feal cipher family," *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 627–638.

[S94] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 191–204.

[W93] Wiener, M., 1993, "Efficient DES key search," presentation at Rump Session of Crypto (August, 1993), Santa Barbara, CA. Available as TR-244, School of Computer Science, Carleton University, Ottawa, Canada, May 1994.

On the Resistance of the CAST Encryption Algorithm to Differential Cryptanalysis

J. Lee[†], H. M. Heys[‡] and S. E. Tavares[†]

[†]Department of Electrical and Computer Engineering
Queen's University
Kingston, Ontario, Canada K7L 3N6

[‡]Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John's, Newfoundland, Canada A1B 3X5

email: tavares@ee.queensu.ca

Abstract

Differential cryptanalysis is a powerful methodology for attacking private-key block ciphers. It has been applied successfully to many ciphers, including FEAL and Khafre. In this paper, it is shown that when randomly generated substitution boxes (s-boxes) are used in the CAST encryption algorithm, the resulting cipher is resistant to the differential attack. Specifically, assuming independent round keys, it is shown that 10 rounds of CAST will provide a better degree of resistance to differential cryptanalysis than 16 rounds of the Data Encryption Standard (DES).

1 Introduction

CAST [1, 2] is a 64-bit private-key block cipher which encrypts by using a number of rounds consisting of large substitution boxes (s-boxes) with fewer input bits than output bits. Two of the most powerful cryptanalytic attacks against iterated block ciphers such as DES [3] and CAST are linear cryptanalysis [4] and differential cryptanalysis [5]. It has been shown that CAST, using randomly generated s-boxes, is resistant to linear cryptanalysis [6]. In this paper, we examine the resistance of CAST to differential cryptanalysis.

The flow of data between consecutive rounds in CAST is similar to that of DES. Both algorithms implement a round function F which operates on the right half of the data block.

The output of F is XOR'd (XOR = exclusive-or) with the left half of the data block to produce a new left half block and then the left and right half blocks are swapped. However, the two ciphers differ significantly in the implementation of the round function F.

In DES, the round function F expands 32 bits of input data to 48 bits using an expansion table E. The expanded data is then XOR'd with 48 key bits and fed into eight 6x4 s-boxes. The output of the 8 s-boxes are concatenated together and then permuted according to a permutation function P to form the 32 output bits. In CAST, the round function XORs 32 bits of input data with 32 key bits and feeds the result into four 8x32 s-boxes. The 32 output bits of the four s-boxes are XOR'd together to form the 32 output bits of F. The round functions of DES and CAST are shown in Figure 1. In [2, 7], CAST is implemented using s-boxes based on partially bent functions. In addition, a key scheduling algorithm is used to assign keys to the various rounds. In this paper, we assume that CAST employs randomly generated s-boxes and uses independent keys in each round of substitution.

2 Distribution of Entries in the XOR Table

Differential cryptanalysis is a chosen plaintext attack which makes use of the highly probable occurrences of sequences of XOR differences at each round given a particular plaintext difference. The foundation for the differential attack is the ability to predict the ouput XOR difference of the round function F given the knowledge of the input XOR difference to that round. Information on the likelihood of possible output XOR values given particular input XOR values is available in an XOR difference distribution table [5]. In the XOR table, each row corresponds to a particular input XOR value, each column corresponds to a particular output XOR value, and the entries themselves represent the

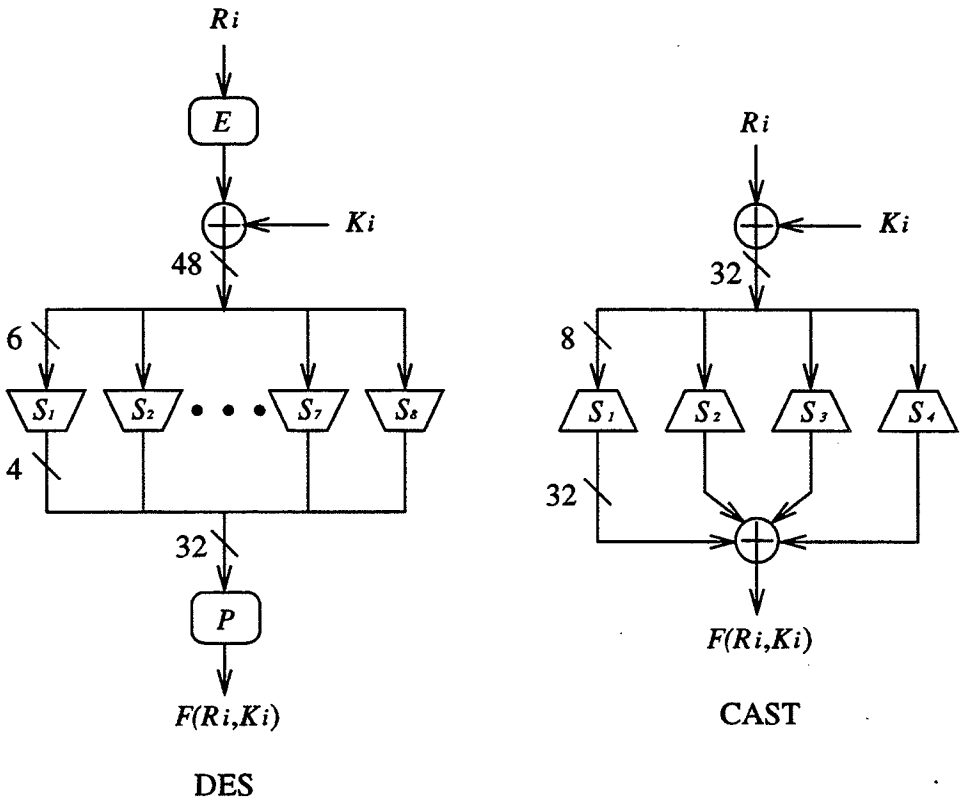

Figure 1: Round Functions of DES and CAST

number of possible pairs corresponding to the input and output XOR value. Using the XOR table, it may be possible to find a highly probable sequence of differences which can be used to cryptanalyse a block cipher.

In reference to CAST, consider a set of four 8x32 s-boxes where we denote the 8-bit inputs to the 4 s-boxes as x_1, x_2, x_3, and x_4, and the corresponding outputs of the 4 s-boxes as $S_1(x_1), S_2(x_2), S_3(x_3)$, and $S_4(x_4)$. For the complete 32-bit round function, given an input XOR value $\Delta x = [\Delta x_1, \Delta x_2, \Delta x_3, \Delta x_4]$, the output XOR value, Δw, is given by the following equation:

$$\Delta w = \bigoplus_{i=1}^{4} [S_i(x_i) \oplus S_i(x_i^*)] \tag{1}$$

where $\oplus$ is the bitwise XOR operation and $x_i^* = x_i \oplus \Delta x_i$.

Define the following functions:

$$f(\Delta x_i) = \begin{cases} 0 & \text{if } \Delta x_i = 0 \\ 1 & \text{if } \Delta x_i \neq 0 \end{cases} \tag{2}$$

and

$$g(\Delta x) = \sum_{i=1}^{4} f(\Delta x_i). \tag{3}$$

Thus, $g(\Delta x)$ is the number of s-boxes that have non-zero XOR inputs when Δx is applied. In fact, if $\Delta x_i \neq 0$ for $i = 1$ to 4, i.e. , $g(\Delta x) = 4$, then all the entries in the XOR table corresponding to that Δx will be multiples of 16. This means that the entries will be 16, 32, 48, etc. . By manipulating equation (1), one can see that all the following input pairs have the same input XOR value $\Delta x = [\Delta x_1, \Delta x_2, \Delta x_3, \Delta x_4]$ and the same output XOR value Δw:

$$[x_1, x_2, x_3, x_4] \leftrightarrow [x_1^*, x_2^*, x_3^*, x_4^*]$$

$$[x_1^*, x_2, x_3, x_4] \leftrightarrow [x_1, x_2^*, x_3^*, x_4^*]$$

$$[x_1, x_2^*, x_3, x_4] \leftrightarrow [x_1^*, x_2, x_3^*, x_4^*]$$

$$[x_1, x_2, x_3^*, x_4] \leftrightarrow [x_1^*, x_2^*, x_3, x_4^*]$$

$$[x_1, x_2, x_3, x_4^*] \leftrightarrow [x_1^*, x_2^*, x_3^*, x_4]$$

$$[x_1^*, x_2^*, x_3, x_4] \leftrightarrow [x_1, x_2, x_3^*, x_4^*]$$

$$[x_1^*, x_2, x_3^*, x_4] \leftrightarrow [x_1, x_2^*, x_3, x_4^*]$$

$$[x_1, x_2^*, x_3^*, x_4] \leftrightarrow [x_1^*, x_2, x_3, x_4^*]$$

Since each of the above pairs constitutes an entry of 2 in the XOR table corresponding to the row Δx and the column Δw, 8 pairs will give rise to an entry of 16 in the XOR table.

Suppose it happens that for the same Δx, there also exists an input $\hat{x} = [\hat{x_1}, \hat{x_1}, \hat{x_3}, \hat{x_4}]$, different from all the 16 inputs listed above, such that the output XOR is also Δw, then the entries in the XOR table corresponding to the row Δx and the column Δw will be 32. Entries of 48, 64, etc. can be explained in a similar way.

In order to determine the likelihood of XOR table entries of 16, 32, etc. , we can rewrite equation (1) as:

$$\Delta w = \bigoplus_{i=1}^{4} \Delta y_i \tag{4}$$

where $\Delta y_i = S_i(x_i) \oplus S_i(x_i^*)$. Since Δw is 32 bits long, it can assume at most 2^{32} distinct values. However, each Δy_i can assume at most 2^7 values for a particular Δx_i. This occurs because for a fixed Δx_i, since x_i is 8 bits long, there will be $2^8/2 = 2^7$ unordered pairs of $(x_i, x_i \oplus \Delta x_i)$. If each of these pairs gives rise to a distinct value for Δy_i, then Δy_i can take at most 2^7 distinct values.

Since the output vectors of the s-boxes are randomly generated, the values obtained by the modulo 2 sum of the Δy_i 's will also be randomly distributed. This results because the j-th bit of the output XOR, $\Delta w^{(j)}$, is just the modulo 2 sum of the j-th bit of the four s-box XOR outputs, $\bigoplus_{i=1}^{4} \Delta y_i^{(j)}$. Since the output bits are randomly generated, it follows that each output XOR bit of an s-box has an equal chance of being 0 or 1. Assuming independence between the output XOR bits of different s-boxes, the modulo 2 sum of the j-th bit of the four s-box output XORs will also have an equal chance of being 0 or 1. Consequently, one can conclude that Δw may assume any one of the 2^{32} possible values with equal probability.

In fact, as the possible values of Δw can be found by trying all the $(2^7)^4$ different

combinations of $\Delta y_1 \oplus \Delta y_2 \oplus \Delta y_3 \oplus \Delta y_4$, the distribution of output XORs for a given input XOR is equivalent to tossing $(2^7)^4$ balls randomly into 2^{32} bins with each ball having a weight of 16. We wish to determine the distribution of the balls in the bins.

Let Y_k be a random variable representing the number of bins having k balls when n balls are being tossed randomly into m bins. It has been shown that for large n and m [8, 9],

$$E[Y_k] \approx m\frac{e^{-\frac{n}{m}}}{k!}(\frac{n}{m})^k \tag{5}$$

For $m = 2^{32}$ and $n = 2^{28}$, $E[Y_k]$ will be the expected number of Δw values that have XOR entries of 16*k for a particular Δx when $g(\Delta x) = 4$. By dividing E[Y_k] by m, one can get the expected percentage of Δw values that have XOR entries of 16*k. For choices of Δx such that $g(\Delta x) = 3$, the corresponding entries in the XOR table can be shown to be multiples of 2048 [10] using a similar argument as in the case of $g(\Delta x) = 4$ and, in this case n = 2^{21}. A summary of the results for $g(\Delta x) = 4$ and $g(\Delta x) = 3$ is listed in Table 1.

For Δx's which have $g(\Delta x) = 2$, the corresponding non-zero entries in the XOR table will be multiples of 2^{18}. In fact, the probability that all the entries are either 0 or 2^{18} is practically 1. For those Δx's which have $g(\Delta x) = 1$, the corresponding non-zero entries in the XOR table will be multiples of 2^{25}. The probability of having entries of magnitude $k * 2^{25}$ for k to be an integer greater than 1 is practically zero. Finally, for the trivial case where $g(\Delta x) = 0$ (i.e., $\Delta x \equiv 0$), there is an entry of magnitude 2^{32} for the column corresponding to $\Delta w = 0$ and the entries are zero for all the other columns.

Note that as the value of $g(\Delta x)$ goes down, the corresponding magnitudes of the entries in the XOR table will go up and it is extremely unlikely for a row in the XOR table corresponding to $g(\Delta x) = i$ to have entries that are greater than that of a row corresponding

$g(\Delta x)$	Entry Value	% of Entries
4	0	93.94
4	16	5.87
4	32	0.183
4	48	$3.83 * 10^{-3}$
4	64	$5.97 * 10^{-5}$
4	80	$7.47 * 10^{-7}$
4	96	$7.78 * 10^{-9}$
4	112	$6.94 * 10^{-11}$
⋮	⋮	⋮
3	0	99.95
3	2048	0.0488
3	4096	$1.19 * 10^{-5}$
3	6144	$1.94 * 10^{-9}$
3	8192	$2.37 * 10^{-13}$
⋮	⋮	⋮

Table 1: Expected Distribution of Entry Value in XOR Table for $g(\Delta x)$=4 and $g(\Delta x)$=3

to $g(\Delta x) = j$ where $j < i$.

3 Iterative Characteristics

Input XOR differences of zero to the round function F always lead to output XOR differences of zero with a probability 1. This is called the 1-round trivial characteristic. If such a trivial characteristic appears in every r rounds of encryption and the plaintext is equal to the ciphertext after r rounds of encryption, then we say we have an r-round iterative characteristic.

3.1 2 Round Iterative Characteristics

Let Φ represent a particular 32-bit XOR vector. The flow of data in a 2-round iterative characteristic is shown in Figure 2 and has the following form:

$(\Phi, 0)$

$0 \leftarrow 0$ with probability 1

$0 \leftarrow \Phi$ with probability p

$(\Phi, 0)$

where the elements in brackets represent the left and right half XOR values respectively and the arrow represents the mapping of the round function F.

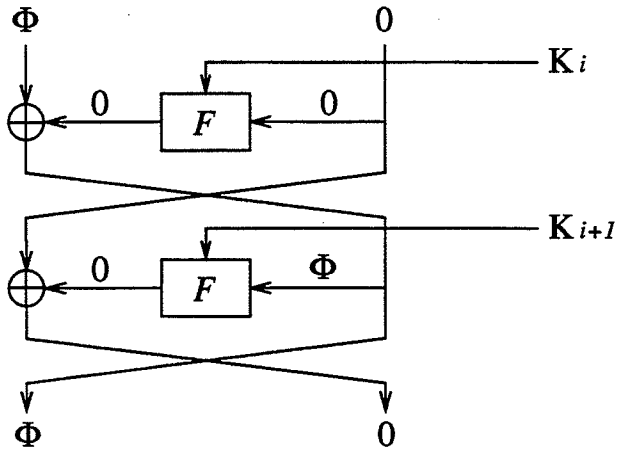

Figure 2: 2-round Iterative Characteristic

Let $q = \mathrm{P}(h(\Delta x, 0) = 16 \mid g(\Delta x) = 4)$ where $h(\Delta x, 0)$ is the entry in the XOR table corresponding to an input XOR value of Δx and an output XOR value of 0. Since Δw is randomly distributed among the 2^{32} possible values, for a particular Δx with $g(\Delta x) = 4$, one can use Table 1 to predict that $q = 0.0587$.

The number of Δx with $g(\Delta x) = 4$ is $(2^8 - 1)^4$. Let μ represent the expected number of rows in the XOR table that will have the entry value $h(\Delta x, 0)$. Thus, for $h(\Delta x, 0) = 16$, $\mu = q * (2^8 - 1)^4 \approx 2.5 * 10^8$ if we assume that the occurrence of $h(\Delta x, 0) = 16$ for different Δx's are independent. An entry of 16 in the XOR table means that the corresponding

differential probability p will be $\frac{16}{2^{32}} = 2^{-28}$.

Using a similar analysis for $g(\Delta x) = 3, 2$ and 1, one can get the values listed in Table 2. It is highly unlikely to find an s-box with which to construct a 2-round iterative characteristic with $p = 2^{-7}$ (on average, 3 out of 100,000 selections). In fact, it would not be difficult to apply a screening process to prevent the occurrence of a 2-round iterative characteristic with a probability of $p = 2^{-7}$. However, it is very likely that an s-box will exhibit a 2-round iterative characteristic with $p = 2^{-14}$ when $g(\Delta x) = 2$. We shall assume that the probability per round for the best 2-round iterative characteristic is thus $(2^{-14})^{\frac{1}{2}} = 2^{-7}$.

$g(\Delta x)$	$h(\Delta x, 0)$	μ	p
4	16	$2.5*10^{8}$	2^{-28}
4	32	$7.8*10^{6}$	2^{-27}
4	48	$1.6*10^{5}$	$2^{-26.4}$
4	64	$2.5*10^{3}$	2^{-26}
4	80	32	$2^{-25.7}$
4	96	$3.3*10^{-1}$	$2^{-25.4}$
⋮	⋮	⋮	⋮
3	2048	$3.2*10^{4}$	2^{-21}
3	4096	7.9	2^{-20}
3	6144	$1.3*10^{-3}$	$2^{-19.4}$
⋮	⋮	⋮	⋮
2	2^{18}	1.5	2^{-14}
2	2^{19}	$2.8*10^{-6}$	2^{-13}
⋮	⋮	⋮	⋮
1	2^{25}	$3.0*10^{-5}$	2^{-7}
⋮	⋮	⋮	⋮

Table 2: Likelihood of Occurrence of 2-round Iterative Characeristic

3.2 Iterative Characteristics with more than 2 Rounds

According to [11], the flow of data in a 3-round iterative characteristic is as follows:

$(\Gamma, 0)$

$0 \leftarrow 0$ with probability 1

$\Phi \leftarrow \Gamma$ with probability p_1

$\Gamma \leftarrow \Phi$ with probability p_2

$(\Phi, 0)$

where Φ, Γ represent 32-bit XOR vectors. Although the inputs and outputs are not the same, one can concatenate this 3-round characteristic with itself with Φ and Γ interchanged to get a 6 round iterative characteristic such that the inputs and the outputs will be the same.

Since the probability per round for the best 2-round iterative characteristic is 2^{-7}, a 3-round iterative characteristic needs to have $p_1 * p_2 > 2^{-21}$ in order to have a better performance than the 2-round iterative characteristic.

A 3-round iterative characteristic is made up of three 1-round characteristic. In section 2, it was shown that the entries in the XOR table depend on the values of $g(\Delta x)$. The most likely maximum entry when $g(\Delta x) = 1$ is 2^{25} and hence the highest one-round difference probability that is likely to occur in such a case is $\frac{2^{25}}{2^{32}} = 2^{-7}$. Similarly, the highest one-round difference probability that is likely to occur when $g(\Delta x) = 2$, 3 and 4 will be 2^{-14}, $2^{-19.4}(= \frac{3*2048}{2^{32}})$ and $2^{-25.2}(= \frac{7*16}{2^{32}})$ respectively.

In order for $p_1 * p_2 > 2^{-21}$, we only need to consider the case when $p_1 = 2^{-7}$ and $p_2 = 2^{-7}$. This means that $g(\Gamma) = 1$ and $g(\Phi) = 1$. If the XOR input value of the round function F is denoted by Δx and the output XOR value by Δw, we can denote $g(\Delta w)$ as the number of s-boxes in the next round that have non-zero XOR inputs when Δw is used as the XOR input. Consider A to be the event the XOR table contains a value of Δw for

which $g(\Delta w) = 1$ given that $g(\Delta x) = 1$. Then it can be shown that, using the assumption of independence between rows in the XOR table, $P(A) = 3.1{*}10^{-2}$ [10]. Hence, the probability of an s-box having a Φ and Γ so that they can be used in round number 2 of the 3-round iterative characteristic is no greater than $3.1 * 10^{-2}$. Therefore, s-boxes which cannot be used in the 3-round iterative characteristics are plentiful and it would be easy to apply a screening process on the s-boxes to ensure that event A does not occur. Hence, there would not be any 3-round iterative characteristics that would have a better probability per round than the 2-round iterative characteristic.

In general, the format for an r-round iterative characteristic would involve a trivial round where the input XOR value of that round is zero, followed by $r-1$ non-trivial rounds. Since the trivial round would have a probability of 1 (zero XOR inputs always give zero XOR outputs), the r-round iterative characteristic would thus have a probability of:

$$p_{\Omega_r} = \prod_{i=1}^{r-1} p_i. \tag{6}$$

where p_i is the probability of the 1 round characteristic in round $i+1$ of the r round characteristic.

The probability per round for the 2-round iterative characteristic is 2^{-7}, and so an r-round iterative characteristic would have a better probability per round than the 2-round iterative characteristic only if $p_{\Omega_r} > (2^{-7})^r$. This implies that $p_i = 2^{-7}$ for $i = 1$ to $r-1$. This results because the next best one-round difference probability is 2^{-14} and the incorporation of just one such XOR difference would make p_{Ω_r} have the same value as $(2^{-7})^r$. Hence we need to have a 1-round difference probability of 2^{-7} for each non-trivial round. This means that for non-trivial rounds, the input pairs differ in at most 1 s-box and

the output pairs differ in at most 1 s-box as well. This is equivalent to event A and the screening process mentioned earlier would ensure that this event does not happen. Hence, an r-round iterative characteristic that gives a better probability per round than the 2-round iterative characteristic will not occur.

4 Differential Cryptanalysis of CAST with R Rounds

One can construct an r round characteristic by concatenating the 2-round iterative characteristic with itself. This r round characteristic will then have a particular plaintext XOR value ΔP and a probability p_{Ω_r} for a particular sequence of XOR values to appear from round 1 to round r. If a plaintext pair having XOR value ΔP does indeed produce the same sequence of XOR values in the intermediate rounds as we would expect from the r round characteristic, then it is a right pair. Otherwise, it is a wrong pair.

By applying the 2-round iterative characteristic and using an $r = R-2$ round attack [5] on a CAST system with R rounds, it can be shown [10] that the probability of a right pair occurring will be:

$$p_{\Omega_{R-2}} = (2^{-14})^{\lfloor \frac{(R-2)}{2} \rfloor} \tag{7}$$

As a result, for an 8 round version of CAST, the highest probability of a right pair occurring is 2^{-42}. CAST constructed with 10 rounds will reduce that probability further to 2^{-56}, a value which is achieved after 16 rounds of DES. For 12 rounds, the probability of the occurrence of a right pair will be 2^{-70}. Since a plaintext block is only 64 bits long, at most we can have 2^{64} different plaintext blocks. A probability of 2^{-70} will thus make it infeasible to apply a successful differential attack on a 12-round CAST cipher.

5 Conclusion

In this paper, a method for predicting the entries in the XOR table of the round function F in CAST using randomly generated s-boxes has been presented. Based on this method, we have shown that the highest probability of predicting an output XOR value given a fixed non-zero input XOR value in one round of encryption is 2^{-7}. The corresponding value in DES is $\frac{1}{4}$. Also, we have shown that in order to apply differential cryptanalysis to an R round CAST system using a simple screening process on the selection of s-boxes for the cipher, the best iterative characteristic is the 2-round iterative characteristic. The best 2-round iterative characteristic has a probability of 2^{-14} and this value is almost 70 times smaller than that of the best 2-round iterative characteristic in DES, which has a probability of $\frac{1}{234}$. As a result, 10 rounds of CAST will reduce the probability of the occurrence of a right pair to 2^{-56}, a value which is better than the 16 rounds of DES encryption.

References

[1] C. M. Adams. *A Formal and Practical Design Procedure for Substitution-Permutation Network Cryptosystems.* PhD thesis, Queen's University, Kingston, Canada, 1990.

[2] C. M. Adams and S. E. Tavares. Designing s-boxes resistant to differential cryptanalysis. In *Proceedings of 3rd Symposium on the State and Progress of Research in Cryptography*, pages 386–397, Rome, Italy, 1994.

[3] *National Bureau of Standards - Data Encryption Standard.* Federal Information Processing Standard Publication 46, 1977.

[4] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology: Proceedings of EUROCRYPT '93*, pages 386–397. Springer-Verlag, Berlin, 1994.

[5] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, vol. 4, no. 1:pages 3–72, 1991.

[6] H. M. Heys and S. E. Tavares. On the security of the CAST encryption algorithm. In *Canadian Conference on Electrical and Computer Engineering*, pages 332–335, Halifax, Nova Scotia, Canada, Sept. 1994.

[7] C. M. Adams. Simple and effective key scheduling for symmetric ciphers. In *SAC '94*, pages 129–133, Kingston, Ontario, Canada, May 1994.

[8] I. Blake. *An Introduction to Applied Probability.* Robert E. Krieger Publishing Company, 1987.

[9] W. Feller. *An Introduction to Probability Theory and its Applications.* New York: Wiley, 1968.

[10] J. Lee and S.E. Tavares. Differential cryptanalysis of CAST cryptosystems using randomly generated s-boxes. Technical report, Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada, Feb 1995.

[11] L. R. Knudsen. *Block Ciphers - Analysis, Design and Applications.* PhD thesis, Aarhus University, Denmark, July 1994.

An average case analysis of a differential attack on a class of SP-networks

Luke O'Connor *

Distributed Systems Technology Centre, and
Information Security Research Center, QUT

Brisbane, Australia

Abstract

We propose a differential attack on tree-structured substitution-permutation networks. The number of chosen plaintexts required for the differential attack to succeed is called the complexity. Our main result is to show that the expected complexity of the attack is linear in the size of the network. This is the first rigorous result concerning the complexity of a differential attack for a general class of product ciphers.

1 Introduction

Differential cryptanalysis [4] has been applied to a considerable number of block ciphers, with great success in most cases. The attack proceeds by finding a series of differences Ω known as a *characteristic*, or a set of related of characteristics known as a *differential*, which then gives information about the secret key used in a cipher, when a sufficient number of plaintext/ciphertext pairs are examined. Assume that we wish to determine the subkey K_R that is being used in round R. The method of differential cryptanalysis can be summarized as

Step 1 Find a highly probable r-round characteristic $\Omega(\Delta P, \Delta C_1, \Delta C_2, \ldots, \Delta C_r)$ which gives (partial) information about the input and output differences of the round mapping F at round R ;

*Correspondence should be sent to DSTC, Level 12 ITE Building, QUT, Gardens Point, 2 George Street, GPO Box 2434, Brisbane Q 4001, Australia; Email oconnor@dstc.edu.au.

Step 2 Uniformly select a plaintext pair P, P' with difference ΔP and encrypt this pair, and assume that P, P' is a right pair in that Ω correctly predicts the ciphertext differences at each round. Determine the candidate subkeys $K'_1, K'_2, \ldots, K'_d$ such that each K'_i could have caused the observed output difference. Increment a counter for each candidate subkey K'_i ;

Step 3 Repeat Step 2 until one subkey K'_R is distinguished as being counted significantly more often than other subkeys. Take K'_R to be the actual subkey.

If P, P' is a right pair then one of the candidate subkeys $K'_1, K'_2, \ldots, K'_d$ is the actual subkey K_R, and K_R will be counted for each right pair. On the other hand, if P, P' is a wrong pair (not a right pair) then we assume that the candidate keys are independent of K_R. So when P, P' is a right pair we record some information about K_R, by incrementing its counter, but record only random information for wrong pairs, by incrementing counters of random keys. The attack succeeds if we can examine enough plaintext/ciphertext pairs so that the true key is counted 'significantly more times' than we expect a random key to be counted. It is then natural to define the *complexity* M of a differential cryptanalysis to be the number of encrypted plaintext pairs of a specified difference required to determine the key or subkey.

One of the shortcomings of the general differential method as described above is the absence of a formula relating the probability p of a right pair to the complexity M of the attack. In general the complexity must be determined by trial and error. The signal-to-noise ratio introduced by Biham and Shamir [3] determines the ratio between the number of times the right key is counted as compared to a random key, which influences the complexity of the attack, but does not formally relate p and M. Miyano [8] has made progress is relating p and M, but his analysis is very heuristic. On the other hand, in linear cryptanalysis [7], for a parity relation of probability p, a data complexity of $|p - \frac{1}{2}|^{-2}$ suffices to determine information about the key with high probability.

The main result of this paper is to display a differential attack on a class of SP-networks devised by Kam and Davida [6]. These SP-networks have the property that each ciphertext bit is provably a function of each plaintext bit, known as completeness or nondegeneracy. We prove (Theorem 4.1) that the expected complexity of our differential attack to recover the keying information is $M = O(Rn/m)$ where $n = m^R$ is the block size and the SP-network is constructed from m-bit bijections. Since the expected value of M is linear in the size of the network, we have a polynomial-time differential attack for all such networks. This result is the first rigorous analysis of a differential attack on a large class of ciphers with no heuristic assumptions.

Several other authors have also indicated weaknesses in the SP-network construction of Kam and Davida. Adams [1] observed that these networks exhibited a poor 'avalanche effect' due to the specific permutations in the construction. Also, Heys and Tavares [5], by extending the work of Anderson [2], have been able to cryptanalyze a larger class of

SP-networks which contain the Kam and Davida networks. However, the attack we will present in this paper requires less the least chosen plaintext to recover the key in a Kam and Davida network.

2 The SP-networks of Kam and Davida

From now on in the paper when we refer to SP-networks, or simply networks, we mean the SP-networks of Kam and Davida [6]. Assume that the network maps n-bit plaintexts $P = p_1, p_2, \ldots, p_n \in Z_2^n$ to n-bit ciphertexts $C = c_1, c_2, \ldots, c_n \in Z_2^n$. The network has R rounds, where each round consists of a substitution followed by a permutation. There are n/m S-boxes S in each round, comprised of two randomly selected m-to-m-bit invertible mappings π_0, π_1, one of which is selected by a key bit (see Figure 1). Such S-boxes will be called *keyed* S-boxes. Let the S-boxes at round r be $S_1^r, S_2^r, \ldots, S_{n/m}^r$ for $1 \leq r \leq R$, such that key bit k_{rj} selects the substitution performed by S_j^r, $1 \leq r \leq R$, $1 \leq j \leq n/m$. The network is constructed to have the following property (proven in [6]).

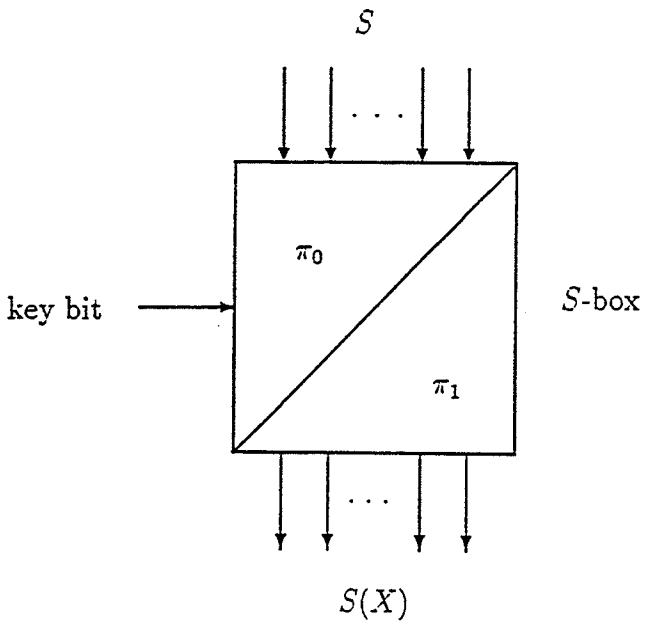

Figure 1: Keyed S-box in an SP-network.

Lemma 2.1 For every S-box S_i^r at round r, $1 \leq r < R$, for the jth input to S_i^r, $1 \leq j \leq m$, there exists a subset of network inputs $A_{ij}^r \subset \{p_1, p_2, \ldots, p_n\}$, $|A_{ij}^r| = m^{r-1}$,

for which complementing any bits in A_{ij}^r only effects the jth input to S_i^r. Moreover, the A_{ij}^r are disjoint, so that $|A_i^r| = m^r$ when $A_i^r = \cup_{1 \leq j \leq m} A_{ij}^r$. □

An example of the resulting network for the parameters $n = 27, m = 3, R = 3$ is given in Figure 2. Observe that in the network in Figure 2, plaintext bit x_1 will enter S-box S_1^1; the outputs of this S-box will spread to the input of 3 distinct S-boxes at round 2; finally at round 3 these outputs will spread to the inputs of 9 S-boxes. It can be shown that after round r, plaintext bit x_i will influence 3^r ciphertext bits. The spread of this influence can be represented as a 3-ary tree of depth 3 having x_i as its root, and all the ciphertext bits c_j as leaves. Anderson [2] has called ciphers which exhibit this type of influence for each variable *tree ciphers*. Essentially, tree ciphers are those ciphers for which the influence of all variables increases geometrically from one round to the next. For this reason, the block size n is of the form $n = m^R$. A recent attack by Heys and Tavares [5] on networks for tree ciphers, extending the initial work of Anderson [2], has a complexity of $M = 2^m \cdot \frac{n-1}{m-1}$. We will prove that a differential method can be used to yield an attack with expected complexity $M = O(Rn/m)$ when the network consists of keyed S-boxes.

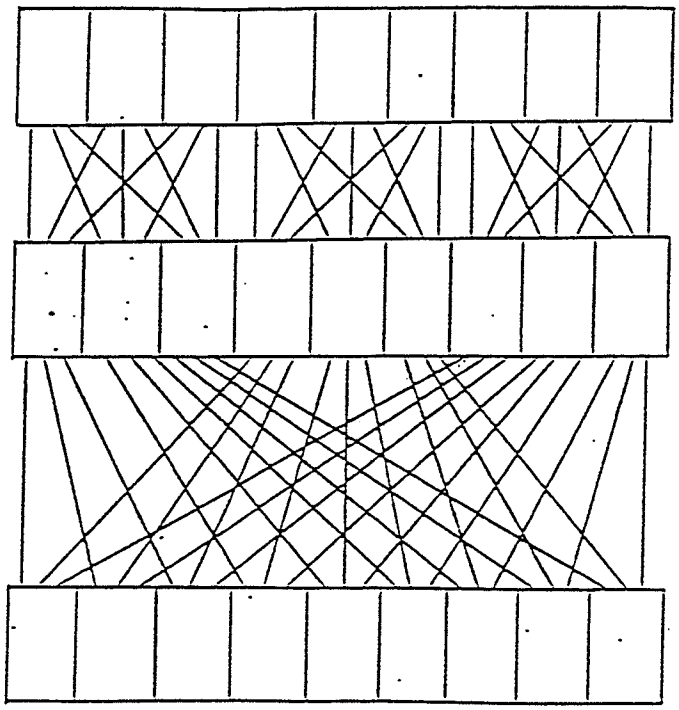

Figure 2: A Kam and Davida SP-network where $n = 27, m = 3, R = 3$.

3 Properties of XOR tables

We will review those notions from differential cryptanalysis [3] that will be required for our purposes. Let $[\cdot]$ be a boolean predicate that evaluates to 0 or 1 such as $[n$ is prime$]$. Let S_{2^m} be the set of all invertible m-bit mappings, which are called m-bit permutations or m-bit bijections. For a given $\pi \in S_{2^m}$ and $\Delta X, \Delta Y \in Z_2^m$, define $\Lambda_\pi(\Delta X, \Delta Y)$ as

$$\Lambda_\pi(\Delta X, \Delta Y) = \sum_{\substack{X, X' \in Z_2^m \\ \Delta X = X + X'}} [\pi(X) + \pi(X') = \Delta Y]. \quad (1)$$

The distribution of $\Lambda_\pi(\Delta X, \Delta Y)$ taken over all possible $\Delta X, \Delta Y \in Z_2^m$ is known as the XOR table for π, denoted as XOR_π.

Example 3.1 For a 3-bit bijection π, let XOR_π be the 8×8 matrix where $\mathrm{XOR}_\pi(i,j) = \Lambda_\pi(i,j)$, $0 \le i,j \le 7$, where i,j are treated as 3-bit binary vectors. Observe that $\mathrm{XOR}_\pi(0,0) = 8$, and all other entries in the first row or column of $\mathrm{XOR}(\pi)$ are zero. If $\pi_0 = (7,2,4,1,5,6,3,0)$ and $\pi_1 = (5,1,7,6,2,4,0,3)$ then

$$\mathrm{XOR}_{\pi_0} = \begin{bmatrix} 8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 4 & \cdot & 4 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 4 & \cdot & \cdot & 4 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 4 & 4 & \cdot \\ \cdot & 2 & 2 & \cdot & 2 & \cdot & \cdot & 2 \\ \cdot & 2 & 2 & \cdot & 2 & \cdot & \cdot & 2 \\ \cdot & 2 & 2 & \cdot & 2 & \cdot & \cdot & 2 \\ \cdot & 2 & 2 & \cdot & 2 & \cdot & \cdot & 2 \end{bmatrix} \quad \mathrm{XOR}_{\pi_1} = \begin{bmatrix} 8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 2 & \cdot & 2 & 2 & \cdot & 2 & \cdot \\ \cdot & \cdot & 4 & \cdot & \cdot & \cdot & \cdot & 4 \\ \cdot & 2 & \cdot & 2 & 2 & \cdot & 2 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 4 & \cdot & 4 \\ \cdot & 2 & \cdot & 2 & 2 & \cdot & 2 & \cdot \\ \cdot & \cdot & 4 & \cdot & \cdot & 4 & \cdot & \cdot \\ \cdot & 2 & \cdot & 2 & 2 & \cdot & 2 & \cdot \end{bmatrix}$$

To emphasize the sparseness of these tables zero entries are represented as a '$\cdot$'. □

The XOR table for an m-bit substitution π has the following general form:

$$\mathrm{XOR}_\pi = \begin{bmatrix} 2^m & 0 & 0 & \cdots & 0 \\ 0 & a_{1,1} & a_{1,2} & \cdots & a_{1,2^m-1} \\ 0 & a_{2,1} & a_{2,2} & \cdots & a_{2,2^m-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & a_{2^m-1,1} & a_{2^m-1,2} & \cdots & a_{2^m-1,2^m-1} \end{bmatrix} = \begin{bmatrix} 2^m & \mathbf{0} \\ \mathbf{0} & A_\pi \end{bmatrix}. \quad (2)$$

For a given mapping π let $\Delta X, \Delta Y$ be an *impossible difference* if the XOR table entry of $\Delta X, \Delta Y$ for π is zero. The following theorem is due to O'Connor [9].

Theorem 3.1 Let π be selected uniformly from S_{2^m}. Then the expected fraction of impossible differences in XOR_π for which $\Delta X \neq 0$ is tending to $e^{-\frac{1}{2}} = 0.6065$. □

Thus approximately 60% of entries in the XOR table for a bijective mapping are impossible when $\Delta X \neq 0$. We will use this property of the XOR tables to distinguish between the two permutations in a keyed S-box.

4 A differential attack

In networks, each S-box S_i is keyed to select one of two possible substitutions $\pi_0, \pi_1 \in S_{2^m}$, that are assumed to be fixed and public for each S-box. For two plaintexts P and P', let X_i and X_i' be the respective inputs entering S_i, such that $\Delta X_i = X_i + X_i'$ is the input difference, and $\Delta Y_i = S(X_i) + S(X_i')$ is the output difference. For sake of illustration let S_i be keyed to implement π_0 so that the XOR table for S_i is actually XOR_{π_0}. Since π_0 and π_1 are independent, Theorem 3.1 states that approximately 60% of the time the entry for the pair $\Delta X_i, \Delta Y_i$ in XOR_{π_1} will be zero (impossible), allowing π_0 to be identified as the keyed substitution. The attack proceeds by repeatedly applying this observation to the S-boxes of the network.

From Lemma 2.1 it is possible to find P and P' such that $w(\Delta X_i) = 1$, where $w(\cdot)$ is the Hamming weight function. Since π_0 and π_1 are invertible, there exists $a, b \in Z_2^m$ such that $\pi_0(a) = \pi_1(b) = Y_i$ and a', b' such that $\pi_0(a') = \pi_1(b') = Y_i'$. If $a + a' \neq b + b'$ then we will say that π_0 and π_1 can be *distinguished* by considering an input difference of ΔX_i. Formally, two substitutions π_0, π_1 can be distinguished if there is a 6-tuple $(\Delta X_i, \Delta Y_i, a, a', b, b')$ such that $\pi_0(a) + \pi_0(a') = \pi_1(b) + \pi_1(b') = \Delta Y_i$ and $a + a' = \Delta X_i$ or $b + b' = \Delta X_i$ but not both.

Example 4.1 Let S_i be a 3-bit S-box such that $\pi_0 = (7,1,0,2,5,6,3,4)$ and $\pi_1 = (3,2,4,7,6,1,5,0)$. If $S_i(X_i) = 4$ then $a = 7$, $b = 2$ since $\pi_0(7) = 4$ and $\pi_1(2) = 4$. If the 2nd bit of X_i can be complemented such that $S(X_i + 010) = 6$, we then determine that $a' = 5$ and $b' = 4$. We conclude that S_i is keyed to implement π_0 since $a + a' = 010$ and $b + b' = 110 \neq 010$. □

The S-box attack outlined above can be directly applied to the S-boxes in the final round of the network, and then to the next to final round and so on, 'peeling off' rounds until all key bits have been determined. At round r let the m inputs to S-box S_i^r be $c_1, c_2, \cdots, c_m$, where $c_j = f_j(p_{1j}, p_{2j}, \ldots, p_{m^{r-1}j})$, since by Lemma 2.1, each input depends on m^{r-1} plaintext bits. Here f_j is a boolean function which describes the dependencies between c_j and the plaintext after $r-1$ rounds. Thus given two inputs that only differ in bit positions p_{ij}, $w(\Delta X_i) \leq 1$ since by Lemma 2.1 none of the other inputs to S_i^r depend on the bi ts p_{ij}. For this reason we define the following notation. If α is an assignment to the m^{r-1} plaintext bits p_{ij}, $1 \leq i \leq m^{r-1}$, then let $\alpha \subseteq P$ denote an assignment to the n plaintext bits p_i where α and P agree for the assignments of th e bits p_{ij}, $1 \leq i \leq m^{r-1}$. Similarly define $\alpha' \subseteq P'$.

Figure 3 shows our attack for finding the key at round r of an network based on distinguishing substitutions, assuming that the keys for rounds $r+1, \ldots, R-1, R$ have already been determined. For each S-box S_i^r at round r, the algorithm searches for two plaintexts P, P' such that $\Delta X_i = X_i + X_i' = e_j$ (the jth unit vector), determines a, a', b, b' as in Example 4.1, and attempts to distinguish between π_0 and π_1.

```
for i ← 1 to n/m do
    distinguished ← false ; j ← 1
    while not distinguished and j ≤ m do
        repeat
            α ← random assignment to p_{1j}, p_{2j}, ..., p_{m^{r-1}j}
            α' ← random assignment to p_{1j}, p_{2j}, ..., p_{m^{r-1}j}
            P ← random assignment to p_1, p_2, ..., p_n such that α ⊆ P
            P' ← such that α' ⊆ P' and P' agrees with P on the assignments
            of the bits outside of α'
            Y_i ← output of S_i^r on encrypting P
            Y_i' ← output of S_i^r on encrypting P'

        until Y_i ≠ Y_i'      { ΔX_i = e_j = X_i + X_i' }

        a ← solve(π_0(a) = Y_i); a' ← solve(π_0(a') = Y_i') ;
        b ← solve(π_1(b) = Y_i) ; b' ← solve(π_1(b') = Y_i') ;

        if a + a' ≠ b + b' then
            distinguished ← true
            if a + a' = e_j then
                    output key bit is 0 for S_i^r
            else output key bit is 1 for S_i^r
        else j ← j + 1
    od
od
```

Figure 3: Distinguishing substitutions at round r.

4.1 Analysis of the differential attack

The success of the attack outlined in Figure 3 depends on two quantities: (i) the number of plaintexts that are required to cause the input of an S-box to complement in one bit position, and (ii) the probability that $S(X_i)$ and $S(X_i')$, $w(\Delta X_i) = 1$, will permit π_0 and π_1 to be distinguished. To answer (i), observe that for any fixed key, the network implements an invertible mapping, even if the number of rounds is truncated at $r < R$. Using this observation and the fact that c_j is described by a balanced function f_j, then after two random assignments to the p_{ij} we expect the output of f_j to be complemented. This implies that the **repeat** loop of Figure 3 will is expected to terminate after a constant number of iterations. Then if substitutions can be distinguished with high probability, only $O(1)$ encryptions will be required to determine the key bit of a given S-box.

To answer (ii), we note that the probability of π_0 and π_1 being distinguished is bounded from below by the probability that $\Delta X_i, \Delta Y_i$ is an impossible difference in XOR_{π_0} or XOR_{π_1}. If S_i is keyed to implement π_{j_0}, $j_0 \in \{0, 1\}$, it then follows that

$$\Lambda_{\pi_{j_0}}(\Delta X_i = X_i + X_i', \Delta Y_i = Y_i + Y_i') \;\geq\; 2. \tag{3}$$

But as π_{j_1}, $j_1 = (j_0 + 1) \bmod 2$, is chosen independently of π_{j_0} then $\Delta X_i, \Delta Y_i$ is an impossible difference in π_{j_1} about 60% of the time from Theorem 3.1. Thus we expect only two differences $\Delta X_{1,j}, \Delta Y_{1,j}$ and $\Delta X_{2,j}, \Delta Y_{2,j}$ to be selected before at least one of these differences is impossible in π_{j_1}. If this is the case then it follows that π_0 and π_1 can be distinguished and the key bit for S_i determined. We can now prove the following theorem.

Theorem 4.1 Assuming all substitutions in the network are selected uniformly from S_{2^m}, then the number of encryptions required to recover the key for an network with Rn/m S-boxes is expected to be $O(Rn/m)$.

Proof. The expected number of encryptions required to generate two plaintexts whose input to a given S-box differs by one bit is a constant. Also, the expected number of encryptions required to distinguish between two independent permutations in any single S-box is expected to be a constant. Since the permutations in each S-box are chosen independently then $O(Rn/m)$ encryptions are required to recover the key. □

Our attack is probabilistic and samples at most m entries from the two XOR tables comprising a substitution, where the m possible input differences are $\Delta X_i, w(\Delta X_i) = 1$. On average, only a constant number of these differences need be examined to distinguish the substitution. Further, in the worst case, the probability that a substitution is *not distinguished* after considering all m such input differences is approximately $(1 - e^{-\frac{1}{2}})^m$. In any case, if a particular sample of m XOR table entries fails to distinguish a substitution, the attack can be repeated with new sample entries until the substitution is

distinguished. Also, in practice, the plaintext in Figure 3 could be chosen to maximize the probability of distinguishing the substitutions, rather than at random, since the S-boxes can be studied off-line.

We have generated pairs π_0, π_1 of random m-bit substitutions and attempted to distinguish between these substitutions by using all characteristics ΔX, $w(\Delta X) = 1$. The results of these experiments for $m = 4, 5, 6, 7$ are given in Table 1. For each value of m, 30,000 pairs of substitutions π_0, π_1 were generated, and let Π be the set of such pairs of substitutions. The first two columns of the table are defined as

$$\Pr(\text{not distinguish}) = \frac{1}{|\Pi|} \cdot \sum_{\pi_0,\pi_1 \in \Pi} \sum_{w(\Delta X)=1} \frac{[a + a' = b + b']}{m \cdot 2^m} \tag{4}$$

$$\max \Pr(\text{not distinguish}) = \max_{\substack{\pi_0,\pi_1 \in \Pi \\ w(\Delta X)=1}} \frac{[a + a' = b + b']}{m \cdot 2^m}. \tag{5}$$

Then Pr(not distinguish) is the probability that a random pair of substitutions will not be distinguished for some $w(\Delta X) = 1$. The third column gives the fraction of the XOR table that is zero which is approaching 0.6065 as expected from Theorem 3.1. Also, the fourth column gives the probability that exactly one of $\Lambda_{\pi_0}(\Delta X, \Delta Y)$ and $\Lambda_{\pi_1}(\Delta X, \Delta Y)$ is zero, which is approaching $2 \cdot (1 - e^{-\frac{1}{2}}) \cdot e^{-\frac{1}{2}} = 0.4773$, also as expected by Theorem 3.1. Thus the probability of distinguishing the substitutions is high.

m	Pr(not distinguish)	max Pr(not distinguish)	$\Lambda_{m,0}$	one zero
4	0.0666	0.7500	0.5873	0.4848
5	0.0323	0.3750	0.5968	0.4812
6	0.0158	0.1875	0.6017	0.4792
7	0.0078	0.1093	0.6041	0.4783

Table 1: Properties of XOR table for random m-bit substitutions.

5 Conclusion

Differential cryptanalysis is a method for recovering the key associated with an iterated mapping using differences. In general it is difficult to determine the expected complexity of a differential attack against a general class of product ciphers. We have presented a differential attack on the SP-networks devised by Kam and Davida, and have also been able to provide a rigorous analysis of the expected complexity of this attack.

Kam and Davida [6] have also proposed a variant of the basic SP-network where in addition to the substitution and permutation at each round, a key vector is XORed with the ciphertext before the substitution. Unfortunately this variant is also susceptible to a differential attack. Let K_i be the portion of the key vector that is XORed with the inputs to S-box S_i. If $\Delta X_i = X_i + X_i'$ are the inputs to S_i, then $X_i + K_i$ and $X_i' + K_i$ will be the actual inputs to S_i, and the difference of the inputs will still be ΔX_i. Thus the previously outlined attack can be applied to distinguish substitutions. Once the substitution that an S-box implements has been determined, the key vector K_i to an S-box S_i can be recovered using similar differential methods that have been applied to the S-boxes of DES [3].

References

[1] C. M. Adams. *A formal and practical design procedure for Substitution-Permutation network cryptosystem.* PhD thesis, Department of Electrical Engineering, Queen's Univeristy at Kingston, 1990.

[2] R. J. Anderson. Tree functions and cipher systems. *Cryptologia*, XV(3):194–202, 1991.

[3] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[4] E. Biham and A. Shamir. *Differential cryptanalysis of Data Encryption Standard.* Springer–Verlag, 1993.

[5] H. M. Heys and S. E. Tavares. Cryptanalysis of tree-structured substitution-permutation networks. *IEE Electronics Letters*, 29(1):40–41, 1993.

[6] J. B. Kam and G. I. Davida. A structured design of substitution-permutation encryption networks. *IEEE Transactions on Computers*, 28(10):747–753, 1979.

[7] M. Matsui. Linear cryptanalysis method for DES cipher. *Advances in Cryptology, EUROCRYPT 93, Lecture Notes in Computer Science, vol. 765, T. Helleseth ed., Springer-Verlag*, pages 386–397, 1994.

[8] H. Miyano. A method to estimate the number of ciphertext pairs for differential cryptanalysis. *Advances in Cryptology, ASIACRYPT 91, Lecture Notes in Computer Science, vol. 739, H. Imai et al. ed., Springer-Verlag*, pages 51–58, 1993.

[9] L. J. O'Connor. On the distribution of characteristics in bijective mappings. *Advances in Cryptology, EUROCRYPT 93, Lecture Notes in Computer Science, vol. 765, T. Helleseth ed., Springer-Verlag*, pages 360–370, 1994.