# An Algorithm to Analyze Block Cipher Resistance to Linear and Differential Cryptanalysis

Kashif Ali and Howard M. Heys
Electrical and Computer Engineering,
Memorial University of Newfoundland,
St. John's, NL, Canada, A1B3X5.
Email: {alikashif, howard}@engr.mun.ca.

## Abstract

In this paper, we propose a practical algorithm that can be used to analyze the block cipher structure known as a Substitution Permutation Network (SPN) with respect to linear and differential cryptanalysis. The algorithm has been applied to 16-bit ciphers and some realistic 64-bit ciphers based on $8\times8$ and $4\times4$ S-boxes that possess good cryptographic properties. Experimental data is presented in the paper which demonstrates the usefulness of the algorithm in characterizing the security level of realistic SPN block ciphers.

## I Introduction

In this paper, we study an algorithm used to analyze the application of linear and differential cryptanalysis to Substitution and Permutation Networks (SPNs). In many real world applications the attacker of an encryption system can possess a set of plaintexts and ciphertexts. Linear cryptanalysis [1] provides a mechanism to obtain subkey bits from the knowledge of plaintext and ciphertext bits. It is called a known plaintext attack because the attacker has the knowledge of the set of plaintexts and the corresponding ciphertexts [2].The basic principle of the linear cryptanalysis is to statistically analyze the S-boxes and extend the properties of the S-boxes to the entire cipher structure [3].

Unlike linear cryptanalysis, differential cryptanalysis is a chosen plaintext attack [3] [4]. In this approach the attacker chooses plaintext input and tries to inspect the output in order to obtain the subkey bits. In differential cryptanalysis the attacker tries to make use of the highly probable differences into the last round of the cipher corresponding to certain plaintext differences [4].

In [5], Matsui developed an algorithm to find the best linear approximation probability and differential probability for the Data Encryption Standard (DES). In this paper, building on Matsui's algorithm, we propose a practical algorithm that can be used to analyze the block cipher structure known as a Substitution Permutation Network (SPN) with respect to linear and differential cryptanalysis.

The remaining four sections in the paper are organized as follow: In Section II, an original algorithm developed related to the SPN structure is introduced. The algorithm tries to find the optimal linear approximation and differential characteristic related to SPNs. The algorithm provides a tool to analyze the effectiveness of the attacks against SPNs and improve the design and selection of cipher components. In Section III, the pseudo code of the two-round iterative algorithm is presented. In Section IV, the algorithm result for linear cryptanalysis is shown for a simple 16-bit SPN [3] [6] and in Section V, conclusions are drawn relevant to our work.

## II Two-Round Iterative Algorithm.

In this section, we examine a practical two-round iterative algorithm that is used to analyze an SPNs resistance to linear and differential cryptanalysis. However, due to space limitations we do not review the application of linear and differential cryptanalysis to SPNs. The reader is referred to [3]. Before implementing the two-round iterative algorithm we tried a modified version of Matsui's algorithm [5] applied to SPNs. The algorithm proved to be inefficient for SPNs such that a solution for realistically-sized networks of 64-bits and 8 or more rounds could not be found.

In the two-round iterative algorithm, the search for the linear approximation or differential characteristic with the largest bias or probability for the cipher structure is depth first search (DFS) based.

The nature of the use of DFS is to find optimal branches through all layers of the tree such that some branches and nodes do not need to be visited as they are clearly not optimal. This we call *pruning*. Hence pruning tries to reduce search space within the tree structure. When the algorithm is executed an implicit tree structure is formed. Each node in the tree structure is equivalent to $\Gamma X_i$ for linear cryptanalysis and $\Delta X_i$ for differential cryptanalysis, where *i* is the round number of the SPN and $\Gamma X$ represents the linear mask and $\Delta X$ represents the difference of the input to a round [5]. Each layer in the tree structure symbolizes the corresponding round number for the SPN. The purpose of a DFS algorithm is to search for the branches through each layer of the tree structure to find where the optimal solution resides. In the case of linear cryptanalysis, an optimal solution is an approximation with the maximum bias. For differential cryptanalysis an optimal solution is a characteristic with the maximum probability.

The two key aspects that are incorporated into the algorithm are the *greedy approach* and the *intelligent pruning mechanism*. The *greedy approach* in the context of algorithm determines the order of the branches to be searched while the *intelligent pruning mechanism* determines when branches are too bad to be considered for further search in the algorithm based on a conditional check. In order to reduce the number of search operations and avoid unnecessary traversing of nodes and branches within the tree structure, the greedy approach in combination with the intelligent pruning mechanism is used to find the best bias or differential probability of the network as quickly as possible.

The greedy approach applied to our algorithm basically considers the arrangement of these masking/difference output bits from the S-box with respect to the masking/difference input bits, and the active S-box involved during such transformation. If the masked/difference output bits lead to the right solution quickly, then the solution is achieved with lesser number of search operations. Thus in our algorithm we try to find the appropriate $(\Gamma X_i, \Gamma Y_i)$ value from the bias-table or $(\Delta X_i, \Delta Y_i)$ value from the difference-table for each round *i* using a greedy approach, so that the resulting solution of *n* rounds is the best one. The greedy approach is useful because it helps to find the maximum bias by minimizing active S-boxes (based on the hamming weight of $\Gamma Y$ or $\Delta Y$ ) and maximizing S-box bias value [3].

The intuition behind the approach is to collect a list of close-to-optimal results after every two rounds (i.e., the list of results obtained after round 2, are used to calculate good results after round 4). The process is iteratively repeated after every two rounds until the good result is obtained after *n* rounds. The assumption in the approach is that by basing the result for *n*, on good results for *n*-2, the overall result will be good, if not optimal. The results obtained using this approach has been found to give good biases or probabilities for an *n* round approximation or differential.

## III Pseudo code for Two-Round Iterative Algorithm

The pseudo code in Figure 1 and the notation used in the two round iterative algorithm are shown with respect linear cryptanalysis. Similar concepts can be applied to differential cryptanalysis. The algorithm produces as output $\overline{B_n}$ , the estimate of the best *n*-round bias, and the arrays *C* and *D* that contain a list of good *n*-round biases and their corresponding $\Gamma X$ , $\Gamma Y$ values for each round. The notations used explicitly for the pseudo code shown in Figure 1 are discussed below:

- Variable *n* is the number of rounds in the linear approximation of the cipher structure.
- $p_i = (\Gamma X_i, \Gamma Y_i)$ is the bias of the $i^{th}$ round computed using the piling-up lemma [3] [5] . It is based on biases from the active S-boxes.
- $[p_1, p_2, p_3, ....p_n]$ is the *n*-round bias computed using the piling-up lemma [3] [5] .
- $perm(\ )$ represents the permutation of bits as determined by the SPN structure for the given input vector.
- $p_n = \max_{\Gamma Y}(\Gamma X_n, \Gamma Y)$ is the maximum bias over all possible value of $\Gamma Y$ for the given $\Gamma X_n$ value.
- Term $\overline{B_n}$ is the current estimate of the maximum *n*-round bias of the cipher.

Two-Round Iterative Algorithm:

**BEGIN**
$i$=0, $\overline{B_n}$ =0
**FOR** $1 \le k \le n$ **DO**
  **FOR** $1 \le j \le L$ **DO**
    $C_k[j] = 0$
  **END FOR**
**END FOR**
**WHILE** $i < n$
  **IF** $(i == 0)$ **THEN** call Procedure FirstRound ()
  **IF** $(i+2) > n$ **THEN** call Procedure Round ($n$)
  **IF** $(i+2) == n$ **THEN** call Procedure Round-even ($n$-1)
  **ELSE** call Procedure Round ($i$+1)
  $i = i + 2$
**END WHILE**
**EXIT**

Procedure FirstRound ()

**FOR** each candidate for $\Gamma Y_1$ **DO**
  $p_1 = \max_{\Gamma X}(\Gamma X, \Gamma Y_1)$
  $\Gamma X_1 = \Gamma X \mid p_1 = (\Gamma X, \Gamma Y_1)$
  **IF** $[p_1, Z_{n-1}] \ge \overline{B_n}$ **THEN**
    $\Gamma X_2 = perm(\Gamma Y_1)$
    **FOR** each candidate for $\Gamma Y_2$ **DO**
      $p_2 = (\Gamma X_2, \Gamma Y_2)$
      **IF** $[p_1, p_2] \ge$ smallest bias in $C_2$ **THEN**
        INSERT $[p_1, p_2]$ in $C_2$
        INSERT $\Gamma X_1$ in $D_1'$, $\Gamma Y_1$ in $D_1''$
        INSERT $\Gamma X_2$ in $D_2'$, $\Gamma Y_2$ in $D_2''$
      **END IF**
    **END FOR**
  **END IF**
**END FOR**
**RETURN**

Procedure Round ($i$)

**FOR** $1 \le j \le L$ **DO**
  $\delta = C_{i-1}[j]$, $\Gamma X_i = perm(D_{i-1}''[j])$
  **FOR** each candidate for $\Gamma Y_i$ **DO**
    $p_i = (\Gamma X_i, \Gamma Y_i)$
    **IF** $[\delta, p_i, Z_{n-i}] \ge \overline{B_n}$ **THEN**
      $\Gamma X_{i+1} = perm(\Gamma Y_i)$
      **FOR** each candidate for $\Gamma Y_{i+1}$ **DO**
        $p_{i+1} = (\Gamma X_{i+1}, \Gamma Y_{i+1})$
        **IF** $[\delta, p_i, p_{i+1}] \ge$ smallest bias in $C_{i+1}$ **THEN**
          INSERT $[\delta, p_i, p_{i+1}]$ in $C_{i+1}$
          INSERT $\Gamma X_i$ in $D_i'$, $\Gamma Y_i$ in $D_i''$
          INSERT $\Gamma X_{i+1}$ in $D_{i+1}'$, $\Gamma Y_{i+1}$ in $D_{i+1}''$
        **END IF**
      **END FOR**
    **END IF**
  **END FOR**
**END FOR**
**RETURN**

Procedure Round ($n$)

**FOR** $1 \le j \le L$ **DO**
  $\delta = C_{n-1}[j]$, $\Gamma X_n = perm(D_{n-1}"[j])$
  $p_n = \max_{\Gamma Y}(\Gamma X_n, \Gamma Y)$
  $\Gamma Y_n = \Gamma Y \mid p_n = (\Gamma X_n, \Gamma Y)$
  **IF** $[\delta, p_n] \ge \overline{B_n}$ **THEN** $\overline{B_n} = [\delta, p_n]$
  **IF** $[\delta, p_n] \ge$ smallest bias in $C_n$ **THEN**
    INSERT $[\delta, p_n]$ in $C_n$
    INSERT $\Gamma X_n$ in $D_n'$, $\Gamma Y_n$ in $D_n"$
  **END IF**
**END FOR**
**RETURN**

Procedure Round-even ($i$)

**FOR** $1 \le j \le L$ **DO**
  $\delta = C_{i-1}[j]$, $\Gamma X_i = D_{i-1}"[j]$
  **FOR** each candidate for $\Gamma Y_i$ **DO**
    $p_i = (\Gamma X_i, \Gamma Y_i)$
    **IF** $[\delta, p_i, Z_1] \ge \overline{B_n}$ **THEN**
      $\Gamma X_{i+1} = perm(\Gamma Y_i)$
      $p_{i+1} = \max_{\Gamma Y}(\Gamma X_{i+1}, \Gamma Y)$
      $\Gamma Y_{i+1} = \Gamma Y \mid p_{i+1} = (\Gamma X_{i+1}, \Gamma Y)$
      **IF** $[\delta, p_i, p_{i+1}] \ge \overline{B_n}$ **THEN** $\overline{B_n} = [\delta, p_i, p_{i+1}]$
      **IF** $[\delta, p_i, p_{i+1}] \ge$ smallest bias in $C_n$ **THEN**
        INSERT $[\delta, p_i, p_{i+1}]$ in $C_n$
        INSERT $\Gamma X_i$ in $D_{n-1}'$, $\Gamma Y_i$ in $D_{n-1}"$
        INSERT $\Gamma X_{i+1}$ in $D_n'$, $\Gamma Y_{i+1}$ in $D_n"$
      **END IF**
    **END IF**
  **END FOR**
**END FOR**
**RETURN**

Figure 1: Pseudo code for two-round iterative algorithm for linear cryptanalysis

- $Z_{n-i}$ is equal to $2^{n-i} \times |\{\max(\Gamma X, \Gamma Y)/2^m\}|$, where $\max(\Gamma X, \Gamma Y)$ is the maximum value in the bias-table and $m$ is the number of output bits of the S-box.
- $C_{i+1}$ is a sorted array of size $L$. The value of $L$ should be large enough to collect good bias for $i$+1 rounds and is ideally large enough to include the partial optimal solution. $C_{i+1}$ is always sorted in ascending order. If the array is full and the new value is larger than smallest value in array, then the new value is inserted into the appropriate position within the array.
- $D_{i+1}'$ is an array of size $L$. The array collects the masking input values for active S-boxes of round $i$+1 corresponding to probabilities in array $C_{i+1}$.
- $D_i'$ is an array of size $L$. It collects the masking input values for active S-boxes of round $i$. These values help in backtracking the masking inputs and the active S-boxes when the $n$-round probability is found and the corresponding linear approximation is to be determined.
- $D_i"$ and $D_{i+1}"$ represents output masking value corresponding to $D_i'$ and $D_{i+1}'$ respectively.
- INSERT operation inserts the value, if appropriate, into the appropriate position in the array.

Note that the candidates for $\Gamma Y_i$ and $\Gamma Y_{i+1}$ are restricted to scenarios where there are 3 or fewer active S-boxes. Experiment results suggest that the optimal solution satisfies this constraint. This restriction is important or the search space in the DFS approach becomes too large for the search to be practical.

## IV Algorithm Results

A sample result for the algorithm is shown in Table 1 of a 9 round linear approximation of the 16-bit SPN using $4\times4$ S-boxes [3]. The S-box used in the simulation is the DES S-box used in [3]. All the values in Table 1 are in hexadecimal format. In Table 1, each cell corresponds to an S-box. All the non-zero cells depict the active S-boxes in the cipher structure. The value in each cell represents the input to the corresponding S-box. For example, the number 6 present in first row and third column, represents the input value 6 in hexadecimal for the third S-box in round one. The largest bias value $\overline{B_9}$ found by the algorithm is 1.466e-4. Therefore according to [1], about 4.65e+7 known plaintext /ciphertext pairs are required to attack the cipher structure using linear cryptanalysis. Table 1 is useful in viewing the active S-boxes and the actual input to the S-boxes that could be used to formulate an attack. For example, only 16 S-boxes form the linear approximation out of maximum 36 S-boxes. Likewise, a table can be formulated by running the algorithm for differential cryptanalysis. In the case of differential cryptanalysis $\overline{B_9}$ will represent the differential probability of 9 rounds of cipher. Similarly, the algorithm can be applied to 64-bit SPN ciphers.

Table 1

| | | | |
|---|---|---|---|
| 0 | 0 | 6 | 0 |
| 0 | 0 | 2 | 2 |
| 3 | 3 | 3 | 0 |
| E | 0 | 0 | E |
| 9 | 9 | 0 | 0 |
| C | 0 | 0 | 0 |
| 0 | 0 | 8 | 0 |
| 2 | 0 | 0 | 0 |
| 8 | 8 | 8 | 0 |

## V Conclusion

The two-round iterative algorithm is a practical algorithm that can be applied to realistic 64-bit SPN structures. By using this algorithm, good solutions which represent linear approximations with the large biases or differential characteristics with the large probabilities are achieved in practical time. Although there is no way mathematically or practically to guarantee that the result attained is the optimal using this algorithm, the algorithm is a useful tool to look into properties of S-boxes and cipher structures that are necessary for good cryptographic resistance to linear and differential cryptanalysis. Some future work can be the application of the algorithm to other cipher structures and 128-bit SPN structures.

## References:


[1] M. Matsui, "Linear cryptanalysis method for DES cipher", *Advances in Cryptology - EUROCRPT '93 (Lecture Notes in Computer Science no. 765)*, Springer-Verlag, pp. 386–397, 1994.

[2] W. Stallings, *Cryptography and Network Security: Principles and Practice (3rd Edition)*. Prentice Hall, 2002.

[3] H. M. Heys, "A Tutorial on Linear and Differential Cryptanalysis", Technical Report CORR 2001-17, Centre for Applied Cryptographic Research, Department of Combinatorics and Optimization, University of Waterloo, Mar. 2001. (Also appears in Cryptologia, vol. XXVI, no. 3, pp. 189-221, 2002.)

[4] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", *Advances in Cryptology -CRYPTO'90 (Lecture Notes in Computer Science no. 537)*, Springer-Verlag, pp. 2–21, 1991.

[5] M.Matsui, "On Correlation Between the Order of S-boxes and the Strength of DES", *Advances in Cryptology-EUROCRPT'94 (Lecture Notes in Computer Science no. 950)*, Springer-Verlag, pp. 366-375, 1995.

[6] H. M. Heys and S. Tavares, "Substitution-permutation networks resistant to differential and linear cryptanalysis", in *Journal of Cryptology,* vol. 9, pp. 1–19, 1996.