Differential Characteristics Search for Basic SPN Cipher

Gozali Harda Kumara (13502066)
Teknik Informatika
Sekolah Tinggi Elektro Informatika
Institut Teknologi Bandung

Abstract

Finding the most probable differential characteristic for a block cipher is often a tedious task if one do it manually, while a brute force search will require $O(2 \cdot 2^n)$ time, where n is the cipher's block size.

In this paper, we will present an algorithm that will search for best differential characteristic of a given Substitution Permutation Network (SPN). This algorithm uses best-first search, and has been applied to 16-bit cipher with 8 rounds. This algorithm uses an assumption that best characteristic will be found on plaintexts that differs exactly one sub-block.

1 Introduction

Differential cryptanalysis, introduced by Biham and Shamir (1991), is a chosen plaintext attack, which means that the attacker could choose a plaintext to encrypt. It exploits the high probability of certain constant differences of plaintext and the corresponding ciphertext in a block cipher.

Let $P' = [P'_1 P'_2 \ldots P'_n]$ and $P'' = [P''_1 P''_2 \ldots P''_n]$ be two plaintext of a certain block cipher which is encrypted to $C' = E(P') = [C'_1 C'_2 \ldots C'_n]$ and $C'' = E(P'') = [C''_1 C''_2 \ldots C''_n]$. Thus, the plaintext difference is defined by $\Delta P = P' \oplus P''$, hence,

$$\Delta P = [P'_1 \oplus P''_1 \;\; P'_2 \oplus P''_2 \;\; \ldots \; P'_n \oplus P''_n]$$

similarly,

$$\Delta C = [C'_1 \oplus C''_1 \;\; C'_2 \oplus C''_2 \;\; \ldots \; C'_n \oplus C''_n]$$

To perform differential cryptanalysis, an attacker must be able to construct a differential characteristic $\Delta P \rightarrow \Delta C$ that is valid with probability p. It must be noted that to construct such differential characteristic, an attacker must try every possible $\Delta P \rightarrow \Delta C$ pairs, and then pick the most probable of them. This task needs to be done carefully with an enormous amount of time.

This paper will present an algorithm that can be used as a tool to search the best differential characteristic of a basic Substitution Permutation Network cipher. The remaining five sections in the paper are organized as follow: In Section 2, a construction of a basic SPN cipher is introduced. The cipher is then used as a target of our search algorithm. In Section 3, we will describe the differential characteristic constructs of our cipher. In Section 4, the algorithm result to search differential characteristic is presented. Section 5 will show our experimental result for the algorithm, and in Section 6, conclusions are drawn relevant to our work.

2 Basic SPN Cipher

Heys (2001) constructed a basic SPN cipher which use a structure proposed by Feistel (1973). His cipher has a fixed substitution and permutation table, and operates in 4 rounds. We will modify that cipher, so it will have a flexible substitution and permutation table, and a variable number of rounds. This cipher takes a 16 bit input block, and illustrated in Figure 1.

2.1 Operations

From Figure 1, we can see that our cipher consists of three basic operations: Substitution in 4-bits sub-blocks., permutation of bit positions, and mixing with subkeys. These three operations will be repeated for n number of rounds.

2.1.1 Substitution

The substitution portion of our cipher breaks the 16-bits block to four 4-bits sub-blocks. Each sub-block then substitutes its input by looking up to 4x4 S-box (i.e. an S-Box with 4-bits inputs and outputs). This S-box can be easily implemented by an array which has sixteen 4-bits values, and indexed by an integer

representing its input bits. Table 1 will shows an example of such S-Box.

Table 1 Example of an S-Box

0	1	2	3	4	5	6	7
E	4	D	1	2	F	B	8
8	9	A	B	C	D	E	F
3	A	6	C	5	9	0	7

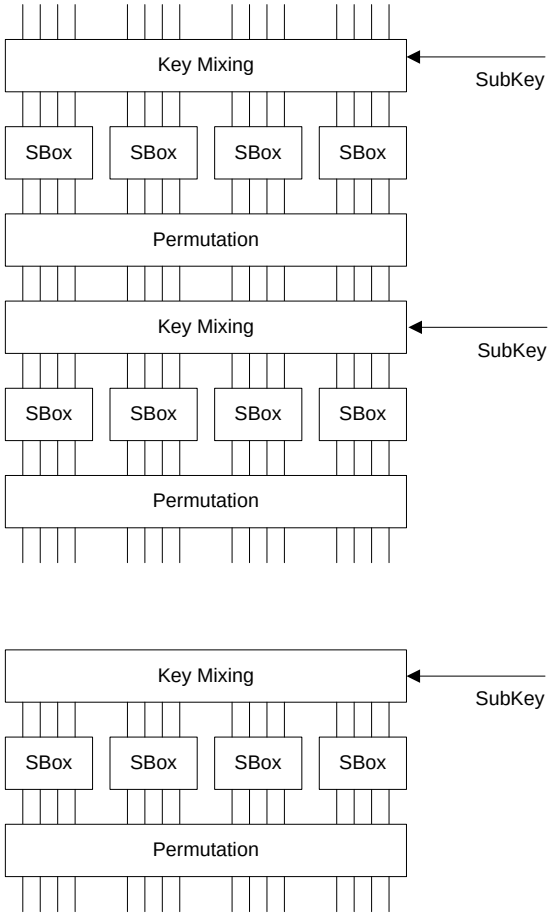


Figure 1. Basic Substitution Permutation Network Cipher

2.1.2 Permutation

After substitutions of sub-blocks have been performed, our cipher combines all of the 4-bits sub-blocks and permutates its bit positions by looking up to P-Box. Our P-Box will take a bit position as its input and outputs the bit's position after permutation has been performed. This P-Box can be implemented by an array which has sixteen integer values representing output positions and indexed by input positions, where 0 corresponds to the left-most bit and 15 to the right-most bit. An example of this P-Box implementation is given in Table 2.

Table 2 Example of a P-Box

0	1	2	3	4	5	6	7
0	4	8	12	1	5	9	13
8	9	10	11	12	13	14	15
2	6	10	14	3	7	11	15

2.1.3 Key Mixing

To perform key mixing, our cipher will generate a subkey for each round, and do bit-wise exclusive-OR between subkey and input block of a round. To get a round-subkey, our cipher will circularly shift its previous round-subkey to the left by one bit. The subkey for the first round is the 16-bit key itself.

2.2 Decryption

In order to decrypt a previously encrypted block, data is passed backwards through the network. However, the decryption's S-Box is the inverse of the encryption's. Thus, in order to allow data to be decrypted, S-Box must be one-to-one mappings with the same number input and output bits.

2.3 Differential Characteristics

To construct a differential characteristic of our cipher, we must first construct differential characteristics for each round. And analyze every operation on them.

2.3.1 Analyzing Substitution Portion

Let $sBox(x) = y$ be a function which performs substitution in our S-Box, and $occurrence(\Delta x \rightarrow \Delta y)$ be the number of occurrence where two inputs with $\Delta x = x' \oplus x''$ difference is substituted by two outputs with $\Delta y = y' \oplus y'' = sBox(x') \oplus sBox(x'')$ difference. Thus, the probability $pSbox(\Delta x \rightarrow \Delta y)$ where $\Delta x \rightarrow \Delta y$ valid can be computed with,

$$pSbox(\Delta x \rightarrow \Delta y) = \frac{occurrence(\Delta x \rightarrow \Delta y)}{16}$$

We know that substations portion of our cipher is performed on four sub-blocks substitutions by S-Box. Thus, if $pSub(\Delta x \rightarrow \Delta y)$ is the probability of Δx substituted by Δy being valid, then,

$$pSub(\Delta x \rightarrow \Delta y)$$

$$= pSBox(\Delta x_0..\Delta x_3 \rightarrow \Delta y_0..\Delta y_3)$$

$$\cup\, pSBox(\Delta x_4..\Delta x_7 \rightarrow \Delta y_4..\Delta y_7)$$

$$\cup\, pSBox(\Delta x_8..\Delta x_{11} \rightarrow \Delta y_8..\Delta y_{11})$$

$$\cup\, pSBox(\Delta x_{12}..\Delta x_{15} \rightarrow \Delta y_{12}..\Delta y_{15})$$

$$= \prod_{i=0}^{3} pSbox(\Delta x_{4i+0}..\Delta x_{4i+3} \rightarrow \Delta y_{4i+0}..\Delta y_{4i+3})$$

2.3.2 Analyzing Key Mixing Portion

In this section, we will prove that key mixing portions of our cipher is irrelevant to differential characteristics.

Let $w' = x' \oplus subkey_i$ and $w'' = x' \oplus subkey_i$ be two key-mixed blocks, thus, their difference Δw is defined by,

$$\begin{aligned}\Delta w &= w' \oplus w'' \\ &= (x' \oplus subkey_i) \oplus (x'' \oplus subkey_i) \\ &= (x' \oplus x'') \oplus subkey_i \oplus subkey_i \\ &= \Delta x\end{aligned}$$

Hence, key-mixed input difference is equal to un-mixed input difference.

2.3.3 Analyzing Permutation Portion

The last portion of our cipher round to analyze is permutation. Let we define $perm^{-1}(y) = x$ to be the inverse function of permutation, thus, by deriving from all of our equation from previous subsection, a round differential characteristics $pRound$ can be computed by,

$$pRound(\Delta x \rightarrow \Delta y) =$$

$$\prod_{i=0}^{3} pSbox\left(\Delta x_{4i+0}..\Delta x_{4i+3} \rightarrow \Delta y_{perm^{-1}(4i+0)}..\Delta y_{perm^{-1}(4i+0)}\right)$$

2.3.4 Characteristics of Overall Cipher

After knowing characteristics of every cipher's round, we can combined them to construct overall cipher differential characteristic.

The overall differential characteristic probability $pCipher$ of our cipher that has $nRounds$ number of round can be computed by,

$$pCipher(\Delta x \rightarrow \Delta y)$$

$$= \prod_{r=0}^{nRounds-1} pRound(\Delta xr_r \rightarrow \Delta xr_{r+1})$$

where $\Delta xr_0 = \Delta x$, and $\Delta xr_{nRounds} = \Delta y$.

3 Search Algorithm

This section will describe our algorithm to search differential characteristic. To construct the algorithm, we must first construct S-Box characteristic, find round characteristic, and finally find cipher characteristic.

3.1 Constructing S-Box Characteristic

Consider an S-Box with input $x = [x_1\, x_2\, x_3\, x_4]$ and output $y = [y_1\, y_2\, y_3\, y_4]$. All its differential characteristics $pSbox(\Delta x \rightarrow \Delta y)$ with probability p can be found by trying every possible input x and x'. The pseudocode of function *fsbc()* in Figure 3 will describe a function that will construct differential characteristics of a given S-Box.

```
Input: SBox: Arrays[0..15] of integer;
Variables: pSBox: arrays [Δx][Δy] of
           probability;
           x, x': 4-bit input;
Initialization: pSBox[Δx][Δy] = 0;
Algoritm:
for (x = 0 to F) do:
  for (x' = 0 to F) do:
    Δx = x ⊕ x';
    Δy = SBox[x] ⊕ Sbox[y'];
    pSBox[Δx][Δy]++;

for each pSBox[Δx][Δy] do:
   pSBox[Δx][Δy] = pSBox[Δx][Δy]/16;

Output: pSBox;
```

Figure 3 Pseudocode of Function *fsbc()*

Function *fsbc()* accepts an S-Box *SBox* as input and will output *pSBox*, where *pSBox[Δx][Δy]* is a probability that $Sbox(\Delta x \rightarrow \Delta y)$ being valid. The function starts with trying to count occurrence of $\Delta x \rightarrow \Delta y$ with x from 0 to F and x' from 0 to F. It then divide the occurrences with 16.

3.2 Finding Round Characteristics

After constructing the S-Box characteristic, we can now find characteristics for a round that has Δx as its input difference. The pseudocode of function *frc()* in Figure 4 will describe a function that will output a list of differential characteristics of a given input difference for a round.

```
Input: Δx: Input Difference;

Variables:
PBox   : Permutation Box;
characs: List of (Δy, probability)
         pairs;
Algoritm:
rsb(characs, Δx, 0, 1, 0);
for each Δy in characs do:
   Δy = permutate(PBox, Δy);

Output: characs;
```

Figure 4 Pseudocode of Function *frc()*

```
Input:
characs: List of (Δy, probability)
         pairs;
Δx, Δy : Input and output difference;
p      : Current probability;
i      : Current Round;

Variables:
SBox   : Substitution Box;
pSBox  : SBox Characteristics;
nr     : Number of Rounds

Algorithm:
if (i = nr) then :
   characs.add(Δy,p x pSBox[Δx_i][Δy_i]);
else :
  if (Δx_i = 0) then :
    Δy_i = Δx_i;
    rsb(characs, Δx, Δy, p, i+1);
  else :
    for (y = 0 to 15) do :
      if (pSBox[Δx_i][y] != 0) then :
        Δy_i = y;
        rsb (characs, Δx, Δy
             p*pSBox[x][y], i+1);
```

Figure 5 Pseudocode of Procedure *rsb()*

Function *frc()* accepts an input block difference Δx as input and will output *characs*, a list of $(\Delta y, p)$ pair where p is probability that $Round(\Delta x \rightarrow \Delta y)$ being valid. Function *frc()* will uses a recurcive procedure rsb() that will use S-Box characteristics for every Δx sub-block. Pseudocode of procedure *rsb()* is described in Figure 5

Procedure *rsb*() is used to find characteristic for every sub-block and works as follow: it first checks whether all sub-block has been processed (this check act as a base for the recursion), if yes, it then add the characteristic to *characs* list. But if all sub-block has not been processed, then it processed current sub-block by checking all possible sub-block characteristics, adding the sub-block output to Δy, multiplying the probability, and then recurse for the next sub-block.

Back to function *frc()*, after running *rsb()*, it permutates all Δy in *characs* with *PBox.* After that, *characs* can be returned as output.

3.3 Finding Cipher Characteristics

Our algorithm to find differential characteristics of basic SPN Cipher will use an assumption that the best characteristics will be found in plaintexts which differ on only one sub-block. Thus, by our definition of differential characteristics at overall cipher, we can build 4x16 trees of differential characteristics and start searching the best characteristic.

It must be noted that, if we transverse the trees using bread-first search, we will need $O(64.16^n)$ space and time at worst-case where n is the number of rounds. We can reduce this by using best-first search where the heuristic function h is defined by,

$$h(node) = pSbox(node.parent.x \rightarrow node.x)^{sb(node.x)}$$

where $sb(x)$ is the number of sub-blocks affected if we permutate x. To reduce our search space even further, we can prune every node where,

$$currentP \,.\, pSbox(node.parent.x \rightarrow node.x)$$

$$\leq$$

$$bestP$$

with $currentP$ is the probability of the node parent, and $bestP$ is the best characteristics probability that has been found. We can prune the tree safely because,

$$pa \,.\, pb \leq pa$$

for $0 \leq pa \leq 1, 0 \leq pb \leq 1$.

Figure 6 will describe the trees that will be constructed. The pseudocode of function *fcc()* that will be used to search that tree will be shown in Figure 7. That function will search the best characteristics of a given cipher.

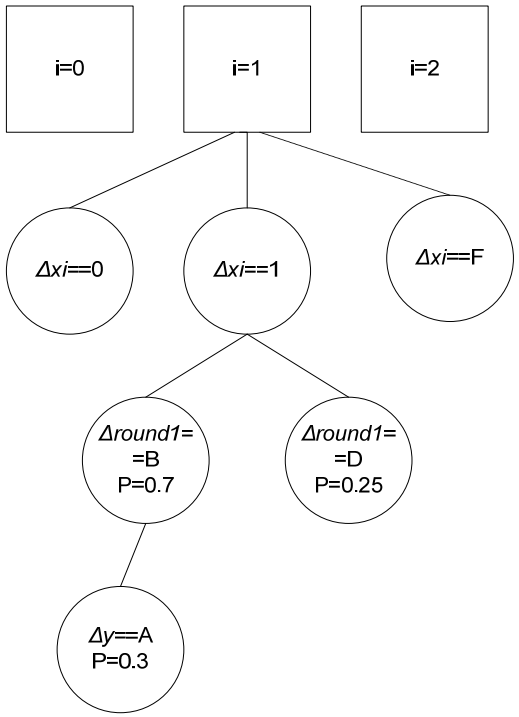


Figure 6 Trees That will be Used to Search Characteristic

Function *fcc()* accepts cipher *Cipher* which is a tupple of (SBox, PBox, NumberOfRound) as input and will output *best*, the best characteristic of *Cipher*. A characteristic is a tuple of $(\Delta x, \Delta round, \Delta y, p)$, which means a differential characteristic $\Delta x \rightarrow \Delta y$ that is valid with probability p, and for every round i has $\Delta round_i$ as its input.

Function *fcc()* will uses a recursive procedure *rr()* that will recurse searching characteristic for the number of rounds. The pseudocode procedure rr() is described in Figure 8.

```
Input: Cipher: a tupple of
               PBox : Permutation Box;
               SBox : Substitution Box;
               nr   : Number of Rounds;

Variables:
pSBox : SBox Characteristics;
best  : Best Characteristic, tupple of:
        best: tupple of:
              Δx : input difference
              Δround: round inputs
              Δy : output difference
              p  : probability
Algoritm:
pSBox = fsbc(Cipher.SBox)
for i = 0 to 3 do :
  for x = 0 to F do
    Δx = 0 & (x << i*4)
    rr(Δx, 0x0, Cipher.nr, 1);

Output: best;
```

Figure 7 Pseudocode of Function *fcc()*

Procedure *rr*() is a recursive procedure to find characteristic for every round and works as follow: it first find all possible round characteristic of the input using *frc()*, then it checks whether it has remaining rounds (this check act as a base for the recursion), if there is no remaining rounds, it find the best probability from *characs* list and save it to *rc*. It then compare whether *rc* probability is better than *best*, if yes, it replace *best* with *rc.*

But if there is still exist remaining rounds, the procedure will sort *characs* using heuristic function *h* mentioned earlier. Then for every characteristics in *characs* it checks whether the probability is still better than *best,* if yes, it continues to recurse the function to the next round with current round output as the next round input, and with the product of current probability and characteristic's probability as the next round probability.

The recursion is guaranteed to stop because it will stop when there is remaining round and for every recursive call the procedure will decrement the

remaining round. After recursion is done, the *best* characteristic can be returned as *fcc()* output.

```
Input:
c    : Current Characteristic
Δx   : Round Input
i    : Remaining Round;
p    : Current Probability;

Variables:
rc   : Characteristic
tempP: Probability

Algorithm:
characs = frc(Δx);

if (i = 0) then :
   rc = findBest(characs);
   p = rc.p * p
   if (p > best.p) then;
     c.Δy = rc.Δy;
     c.p = p;
     best = c.clone();

else :
  sort(characs);
  for each rc in characs do:
    temp = rc.p * p
    if (p > best.p) then:
     c. ΔRound_i=rc.Δy;
     rr(c, rc.ΔOutput, round-1, tempP)
```

Figure 8 Pseudocode of Procedure *rr()*

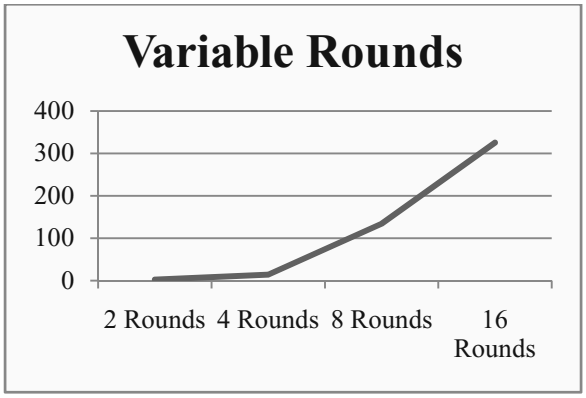
4 Related Works

Ali and Heys (2007) presented an algorithm to analyze the resistance of block cipher to differential and linear cryptanalysis. In their paper, they call their algorithm Two Iterative Way. That algorithm uses greedy based and intelligent pruning. Their algorithm has been applied to 16-bit ciphers and some realistic 64-bit ciphers based on 8x8 and 4x4 S-Boxes that possess good cryptographic properties.

Heys (2001) was described in block cipher design. In his paper, he presented a detailed tutorial on linear and differential cryptanalysis.

5 Experiments and Results

Our algorithm will be implemented as a Java application. A Windows XP SP2 platform with JDK 1.6 on a 1.60 GHz Intel Centrino processor is used for the experiments. The algorithm's running time is measured with Java's `System.nanotime()`. We conducted varied experiment based on round numbers and block sizes. Figure 9 will show the effect of round numbers and block sizes to running time of our algorithm.



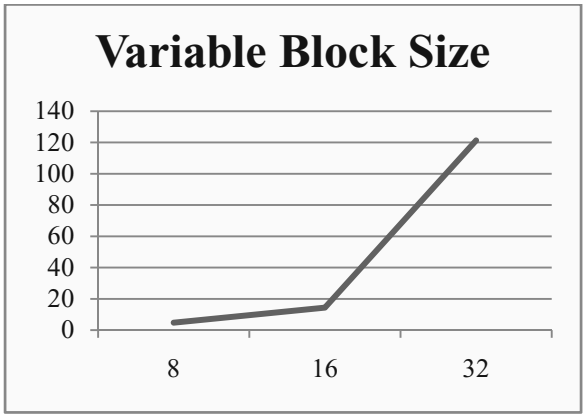


Figure 9 Effect of Algorithm Round Numbers and Block Sizes to Algorithm Running Time

5.1 Variable Round Numbers

On this experiment, we will use ciphers with S-Box and P-Box that is mentioned in Section 2 but with variable round-number. Table 3 will show the result of our experiments.

From the table it is shown that the running time of our algorithm will increase more than quadratic when the round number increased quadratically. The sample of our cipher outputs will be shown in Figure 10.

Table 3 Running Time Algorithm in Varied Round Numbers

Rounds	∆x	∆y	p	Time (s)
2	0x0B00	0x0220	0.25	2.6
4	0x0B00	0xB2B2	0.0264	14.3
6	0x00B0	0x6255	0.0053	43.2
8	0x0400	0xB5F2	0.0018	134.2
16	0x0400	0xC06F	0.000023	325.5

```
> java DiffSearch sbox.txt pbox.txt 4
dx: B00
dx1: 40
dx2: 220
dx3: 550
dy: B2B2
time = 14385 ms
```

Figure 10 Output for 4-Round Cipher

5.2 Variable Block Size

On this experiment, we will use ciphers with S-Box with 4 round. For every cipher, we will use randomized P-Box. Table 3 will show the result of this experiment. From the table it is shown that, like our previous experiment, the running time of our algorithm will increase more than quadratic when the round number increased quadratically.

Table 4 Running Time Algorithm in Varied Block Sizes

Block Size	Time(s)
8	4.7
16	14.3
32	121.3

6 Conclusions

We have presented an algorithm to search differential characteristic of basic SPN Cipher. Best-First Search is practical approach to search the search space of differential characteristic. Although we cannot mathematically or theoretically guarantee that our algorithm will find the best characteristic, because of our assumption that the best characteristic will be found on inputs with plaintexts differs in exactly one block.

Although it is predicted and acceptable, the running time of our algorithm increases more than number of rounds or block sizes.

7 References

Heys, Howard M. "A Tutorial on Linear and Differential Cryptanalysis", *2001.*

Ali, Kashiv and Heys, Howard M. "An Algorithm to Analyze Block Cipher Resistance to Linear and Differential Cryptanalysis", *2007.*

Biham, E and Shamir A. "Differential Cryptanalysis of DES-like Cryptosystems", *1991.*

Feistel, H. "Cryptography and Computer Privacy", *1973.*