Improvements for Finding Impossible Differentials of Block Cipher Structures

Yiyuan Luo[1]* and Xuejia Lai[2,3]**

[1] School of Electronics and Information, Shanghai Dianji University
luoyy@sdju.edu.cn
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[3] Westone Cryptologic Research Center, Beijing 100070, China

Abstract. In this paper we improve Wu and Wang's method for finding impossible differentials of block cipher structures. This improvement is more general than Wu and Wang's method that it can find more impossible differentials with less time. We apply it on Gen-CAST256, Misty, Gen-Skipjack, Four-Cell, Gen-MARS, SMS4, MIBS, Camellia*, LBlock, E2 and SNAKE block ciphers. All impossible differentials discovered by the algorithm are the same as Wu's method. Besides, for the 8-round MIBS block cipher, we find 4 new impossible differentials, which are not listed in Wu and Wang's results. The experiment results show that the improved algorithm can not only find more impossible differentials, but also largely reduce the search time.

Keywords: Cryptanalysis; Impossible differential; Block ciphers;

Mathematical Subject Classification: 94A60

1 Introduction

Impossible differential cryptanalysis, introduced by Biham et al. [4] and Knudsen [17] independently, is a special case of differential cryptanalysis that uses differentials with probability zero to sieve the right keys from the wrong keys. It is one of the most powerful attacks for block ciphers and is considered in many block cipher designs [37,9,11,35,20,19,12,33]. The best cryptanalytic results for some block ciphers are obtained by impossible differential cryptanalysis [8,4]. For example, the currently best attack on the 31-round Skipjack is still the impossible differential cryptanalysis by Biham *et al.* [4]

The key step in impossible differential cryptanalysis of a block cipher is to find the longest impossible differential. Given two variables $x_1, x_2 \in \mathbb{F}_2^n$, the difference of x_1 and x_2 is usually denoted as $\Delta x = x_1 \oplus x_2$. An impossible differential for an n-subblock block cipher is in the form $(\Delta in \nrightarrow_r \Delta out)$ where $\Delta in = (\Delta x_1, \ldots, \Delta x_n)$ and $\Delta out = (\Delta y_1, \ldots, \Delta y_n)$. $(\Delta in \nrightarrow_r \Delta out)$ means the probability of the output difference is Δout after r rounds of a block cipher for an input difference Δin is zero. At the first glance, impossible differentials are obtained manually by observing the block cipher structure. However, since the emergence of impossible differential cryptanalysis, automated techniques for finding impossible differentials have been introduced.

The first automated technique is called the Shrinking method introduced by Biham *et al* [4]. This method is simple but very useful. It only considers truncated differentials whose differences distinguish only between zero and arbitrary nonzero difference. Given a block cipher, the adversary first designs a mini version of this block cipher, which is, scales down the block cipher but preserves

* Yiyuan Luo was supported by NSFC (61402280) and Academic Discipline Project of Shanghai Dianji University (16YSXK04)

** Xuejia Lai was supported by NSFC (61272440, 61472251,U1536101), China Postdoctoral Science Foundation (2013M531174, 2014T70417), National Cryptography Development Fund MMJJ20170105 and Science and Technology on Communication Security Laboratory.

the global structure. Then the adversary exhaustively searches for this mini cipher and obtains some truncated impossible differentials. Usually these truncated impossible differentials of the mini cipher remain impossible differentials in the normal version. This method can deal with most block ciphers in the real world. However, it becomes very slow if the number of subblocks of a block cipher is as large as 16, since exhaustive search on the mini version of this type of cipher is still a heavy load for most computers.

The second automated technique is based on the miss in the middle approach. This method combines two differentials, one from the input and the other from the output, both with probability 1. However, these two differentials cannot meet in the middle since they can never be equal in the middle. The $\mathcal{U}$ method [16,15] and the UID method [23] both belong to this category. In the $\mathcal{U}$ method and the UID method, the adversary first represents the block cipher structure as a matrix, then given a differential pair $(\Delta in, \Delta out)$, he calculates the m-round intermediate difference from Δin forwardly and the $(r-m)$-round intermediate difference from Δout backwardly by the matrix method. If there is a contradiction for these two intermediate differences, then an impossible differential $(\Delta in \nrightarrow_r \Delta out)$ is verified. Representing a block cipher by the matrix has been a popular method in impossible differential, integral and zero correlation linear cryptanalysis [34,21,36,3,33,2,6,19].

In [31], Wu and Wang extend the $\mathcal{U}$-method and UID method to a more generalized method which does not use the miss in the middle approach. They treat the r-round block cipher structure as a system of equations, which describe the propagation behavior of differences in the inner primitives, especially sbox permutations or branch swapping of the block cipher structure. To judge if a truncated differential $(\Delta in, \Delta out)$ is impossible, they predict information about unknown variables from the known ones iteratively. Finally a truncated differential is verified by checking the constrained conditions in the system. This method is similar as a linear programming method for solving optimization problems.

In [28], Sun *et al.* show that Wu and Wang's automatic search method can find all impossible differentials of a cipher that are independent of the choices of the inner primitives. However, Wu and Wang's method can only find all truncated impossible differentials since the choice of truncated difference may result in missing some impossible differentials. Wu and Wang's method only considers differences $\Delta in = (x_1, \ldots, x_n)$ and $\Delta out = (y_1, \ldots, y_n)$ where x_i and y_i are zero or nonzero values. They assign an indicator to indicate the choice of x_i and y_i, representing by 0 a subblock without difference and by 1 a subblock with a difference. The relationships between nonzero differences have been omitted. For example, y_i may be equal to some x_j, where $1 \leq i, j \leq n$. If some linear constraints between nonzero variables in Δin and Δout are needed, Wu and Wang claimed their method could still work by translating all linear constraints into the system of equations. However, this method increases the run complexity and implement of the search method. Since it changes the equation system for every value of $(\Delta in, \Delta out)$ and if the relationship between Δin and Δout is complicated, the matrix will be very large.

The idea of the UID method is it represents the differential with symbols and utilizes the propagation property of the linear accumulated symbols. The idea of the Wu-Wang method is to utilize solving linear equations to determine an impossible differential. We show that the Wu-Wang method can be improved by combining the idea of the UID method and Wu-Wang method. Instead of using 1 to represent the nonzero difference, we use a letter symbol to represent a difference and different symbols represent different nonzero values. This method can represent more relationships between these subblocks. For example, if $\Delta in = (a, 0, 0, a)$ and $\Delta out = (a, 0, 0, b)$ for a 4-subblock structure where a and b are different nonzero values, then we have $x_1 = x_4 = y_1$ and $y_4 \neq x_1$. In our method, the matrix of the system does not need to be changed with $(\Delta in, \Delta out)$. We also improve the Wu-Wang method by simplifying the test of whether there are solutions for linear systems. Since the most time consuming part is the matrix operation, our improved method can find more impossible differentials in less time.

We implement the method in java language and apply it to many block cipher structures, including Gen-CAST256 [34], Gen-Skipjack [29], Four-Cell [10], Gen-MARS [16], Gen-RC6 [29], SMS4 [23], Misty [24], MIBS [13], Camellia* [1], LBlock [32], E2 [14] and SNAKE [18]. For these block ciphers, we rediscover all known impossible differentials. Especially for the 8-round MIBS cipher, we find 4 new impossible differentials, which are not listed in Wu and Wang's work. Our improvement largely

reduced the run time for finding impossible differentials. In [30], the results for MIBS, LBlock and E2 are obtained in a few hours on a 2.66 GHz processor with MAGMA package. However, our results for MIBS, LBlock and E2 are obtained within 10 seconds on a 2.20 GHz processor.

2 Preliminaries

In this section we introduce some basic concepts and notions used in this paper. We first introduce the block cipher structures. Next we review the solvability of a system of linear equations.

2.1 Block Cipher structures

There are two mainly block cipher structures, which are the Feistel structure and its generalizations and the substitute permutation network (SPN). The round function of most of those structures consists of three basic operations: the sbox look-up, the exclusive-or addition (Xor) and the branch swapping, where the only nonlinear component is the sbox look-up operation. In differential cryptanalysis, the Xor differences of plaintext/ciphertext pairs are considered, we omit the key and constant addition since they have no relevance to our analysis. We assume a block cipher structure has n sub-blocks (branches), and the input and output differences are denoted by $(\Delta x_1, \ldots, \Delta x_n)$ and $(\Delta y_1, \ldots, \Delta y_n)$ respectively.

2.2 The solvability of a linear system

Now we review the basics in linear algebra of determining the solvability of a system of linear equations. Let m, n be two positive integers, $m < n$, let $Ax = b$ be a system of m linear equations with n variables, where A is a $m \times n$ matrix over $\mathbb{F}_2$ and $x = (x_1, \ldots, x_n)$ and $b = (b_1, \ldots, b_m)$ are two bit vectors, then the augmented $m \times (n+1)$ matrix $B = [A|b]$ can determine the solvability of the linear system.

A regular method is to deduce the reduced row echelon form (a.k.a. row canonical form) of matrix B by Gauss-Jordan Elimination algorithm. The reduced row echelon form of a matrix is unique and denoted by B'. One start to check B' from the last row to the first, to see if there exist a row which the first n entries are zeros and the last entry is nonzero. If there are such rows, then the linear system has no solution. For example, if the augmented matrix B of a linear system in reduced row echelon form is

$$B' = \begin{pmatrix} 1 & 0 & 0 & b_1 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 0 & b_3 \end{pmatrix}$$

where b_3 is nonzero, then the linear system has no solution.

3 Mathematical Models for Finding IDs of Block Cipher Structures

Our improvement is based on Wu and Wang's method. If the nonlinear sbox S_i in a block cipher structure is a permutation, then there is a constraint on the input difference x_i and output difference y_i for S_i, that is, x_i and y_i can only both be zero or both be nonzero, denote by $x_i \sim y_i$. The intermediate value of a block cipher structure is called the state. The state updates with the round structure. In order to find impossible differential for an r-round block cipher structure, we first set differential variables for the states, then transform the r-round block cipher structure into a system of linear equations and constraints, denoted by $\mathcal{S}$. Then for a given differential $(\Delta in, \Delta out)$ where $\Delta in = (a_1, \ldots, a_n)$ and $\Delta out = (b_1, \ldots, b_n)$, we can check if it is impossible by solving $\mathcal{S}$ with initial values $(a_1, \ldots, a_n, b_1, \ldots, b_n)$, if $\mathcal{S}$ has no solution, then $\Delta in \nrightarrow_r \Delta out$.

Here we take the 5-round Feistel structure as an example. We first assign differential variables for 5-round Feistel structure. In Fig 1, $F_i, 1 \leq i \leq 5$ are permutations, the output difference of F_i

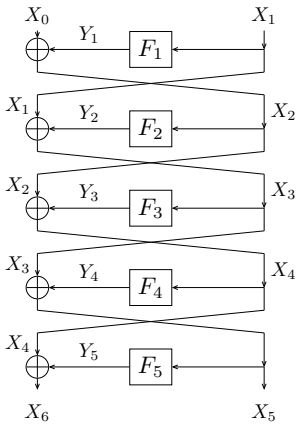

Fig. 1. State variables for 5-round Feistel Structure

for input difference X_i is Y_i, thus $X_i \sim Y_i$. According to the computation graph of 5 round Feistel structure, we obtain the following system $\mathcal{S}$ of equations and constraints:

$$\begin{aligned} X_0 \oplus X_2 \oplus Y_1 &= 0 & X_1 &\sim Y_1 \\ X_1 \oplus X_3 \oplus Y_2 &= 0 & X_2 &\sim Y_2 \\ X_2 \oplus X_4 \oplus Y_3 &= 0 & X_3 &\sim Y_3 \\ X_3 \oplus X_5 \oplus Y_4 &= 0 & X_4 &\sim Y_4 \\ X_4 \oplus X_6 \oplus Y_5 &= 0 & X_5 &\sim Y_5 \end{aligned}$$

In order to check if $(a, 0) \to (a, 0)$ is an impossible differential where a is a nonzero value, we solve the above system with $X_0 = a, X_1 = 0, X_5 = 0, X_6 = a$. Since $X_1 \sim Y_1$ and $X_5 \sim Y_5$ we have $Y_1 = 0$ and $Y_5 = 0$. From linear equations of $\mathcal{S}$, we get $Y_3 = 0$, thus $X_3 = 0$ since $X_3 \sim Y_3$, next from linear equations $\mathcal{S}$ we obtain $Y_2 = 0$, however $X_2 = a$ and $X_2 \sim Y_2$, thus the system $\mathcal{S}$ has no solution and $(a, 0) \to (a, 0)$ is an impossible differential for 5-round Feistel structure.

Now we want to find all impossible differentials for 5-round Feistel structure, we enumerate all the possible differential pairs $(\Delta in, \Delta out) \in \{(0, a), (a, 0), (a, a), (0, b), (b, 0), (b, b), (b, a), (a, b)\}$ where a and b are two different nonzero values. For each value of $(\Delta in, \Delta out)$, we judge if it is an impossible differential, after all cases are tested, we will find all impossible differentials.

Thus the general algorithm for finding all r-round impossible differentials for a block cipher structure is outlined as:

1. Generate all the possible differential pairs $(\Delta in, \Delta out)$ in a set $\mathcal{D}$.
2. Assign differential variables according to the computation figure of the r-round block cipher structure. Generate the system $\mathcal{S}$ of linear equations and constraints with the differential variables.
3. For each $(\Delta in, \Delta out) \in \mathcal{D}$, solve the system $\mathcal{S}$ with initial value $(\Delta in, \Delta out)$ and check if $\mathcal{S}$ has no solution. If there is no solution, then $(\Delta in \to \Delta out)$ is an impossible differential. After all cases are checked we obtain all impossible differentials.

4 The Detailed Algorithm

In this section we describe the detailed algorithm and implementation details.

4.1 Generate all possible differential pairs

We use symbols $a_i, b_i, 1 \le i \le n$ to denote $2n$ different the nonzero values. For a block cipher structure, the input difference is $(\Delta I_1, \ldots, \Delta I_n)$ where $\Delta I_i \in \{0, a_1, \ldots, a_n\}$ and the output difference is

$(\Delta O_1, \ldots, \Delta O_n)$ where $\Delta O_i \in \{0, a_1, \ldots, a_n, b_1, \ldots b_n\}$. Note that the input difference and output difference will not be zero since it will be trivial in differential cryptanalysis. Thus there are total $((n+1)^n - 1) \cdot ((2n+1)^n - 1)$ differential pairs. This value is large for many block cipher structures.

However, an impossible differential $(\Delta I_1, \ldots, \Delta I_n) \nrightarrow (\Delta O_1, \ldots, \Delta O_n)$ for a block cipher structure is usually simple, that is, there are very few nonzero values in $(\Delta I_1, \ldots, \Delta I_n)$ and $(\Delta O_1, \ldots, \Delta O_n)$. Since if the input or output differential are complicate, it will propagate fast due to the round structure of the cipher. Thus it is reasonable to consider simple differential pairs. Actually all the impossible differentials found for block cipher structures in the literature are simple.

In this paper we only consider the input difference $(\Delta I_1, \ldots, \Delta I_n)$ where $\Delta I_i \in \{0, a\}$ and the output difference $(\Delta O_1, \ldots, \Delta O_n)$ where $\Delta O_i \in \{0, a, b\}$. Thus there are total $(2^n - 1)(3^n - 1)$ differential pairs need to be checked.

4.2 Generate the system $\mathcal{S}$

Given a block cipher structure, we first need to draw the computational figure and assign differential variables, as introduced in the analysis of 5-round Feistel structure. This step is varying according to different block cipher structures. However, since most block cipher structures iterate the same round structure for several times, these variables are regular and easy to implement in a computer program. As in the analysis of 5-round Feistel structure, the input difference of a nonlinear permutation is denoted by variable X_i and the output difference is denoted by variable Y_i. Thus if we see a variable Y_i, it must be some output difference of a nonlinear permutation.

For a block cipher structure with r rounds, there are p variables $X_i, 0 \le i \le p$ and q numbers of variables $Y_i, 1 \le i \le q$. The number p and q are determined by the round structure and the round number r. For the r-round Feistel structure, $p = r + 2$ and $q = r$. We first denote all variables in a variable vector as

$$\overline{X} = (X_0, \ldots, X_{p-1}, Y_1, \ldots, Y_q),$$

then linear equations in system $\mathcal{S}$ can be write as $M\overline{X} = \mathbf{0}$ where M is a $kr \times (p+q)$ matrix over F_2 and $\mathbf{0}$ is a $(p+q)$-dimensional zero vector, where k is the number of linear equations in one round of the block cipher structure. The augmented matrix of these linear equations is $B = [M|\mathbf{0}]$. For the 5-round Feistel structure, the augmented matrix B is denoted in Table 1 .

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
	X_0	X_1	X_2	X_3	X_4	X_5	X_6	Y_1	Y_2	Y_3	Y_4	Y_5	**0**
1	1	0	1	0	0	0	0	1	0	0	0	0	0
2	0	1	0	1	0	0	0	0	1	0	0	0	0
3	0	0	1	0	1	0	0	0	0	1	0	0	0
4	0	0	0	1	0	1	0	0	0	0	1	0	0
5	0	0	0	0	1	0	1	0	0	0	0	1	0

Table 1. The 5×13 augmented matrix of 5-round Feistel structure

The set of constraints in $\mathcal{S}$ can be maintained as a map $\mathcal{N}$. Let $id(X_i)$ denotes the index of the variable X_i in vector $\overline{X}$, given a constraint $X_i \sim Y_i$, we add $(id(X_i), id(Y_i))$ into the map $\mathcal{N}$. For the 5-round Feistel structure, $\mathcal{N} = \{< 1, 7 >, < 2, 8 >, < 3, 9 >, < 4, 10 >, < 5, 11 >\}$. In the real implementation, it is noted that for most block cipher structures, the distance between a constraint X_i and Y_i is fixed and determined by the round structure and the round number, that is, $id(Y_i) - id(X_i)$ is a constant. For example, the distance of constraint X_i and Y_i for a r-round Feistel structure is $r+1$. Thus the map $\mathcal{N}$ is not needed to be implemented but only the fixed index distance is needed. This observation facilitates the real implementation of the algorithm.

4.3 Determine the solvability of $\mathcal{S}$

In the beginning, we assign a symbol '?' to each variable in the variable vector $\overline{X}$, which means every variable is undetermined. Given a differential pair $(\Delta in, \Delta out)$, we need to check if there exist solutions of the system $\mathcal{S}$ with the initial value $(\Delta in, \Delta out)$. We first need to initialize the variable vector $\overline{X}$ according to $(\Delta in, \Delta out)$. As in the 5-round Feistel structure, for a differential pair $\Delta in = (a, 0), \Delta out = (a, 0)$, the variable vector $\overline{X}$ is initialized as:

X_0	X_1	X_2	X_3	X_4	X_5	X_6	Y_1	Y_2	Y_3	Y_4	Y_5
a	0	?	?	?	0	a	?	?	?	?	?

For a constraint $X_i \sim Y_i$, the algorithm updates (X_i, Y_i) and detects contradictions as follows.

- If the value X_i is updated,
  - If $X_i = 0$ and $Y_i =?$, then Y_i is set to 0;
  - If X_i is a nonzero symbol and $Y_i =?$, then Y_i is set to the nonzero symbol '*';
  - If $X_i = 0$ and Y_i is an nonzero symbol, then we obtain a contradiction;
- If the value Y_i is updated,
  - If $Y_i = 0$ and $X_i =?$, then X_i is set to 0;
  - If $Y_i = 0$ and X_i is an nonzero symbol, then we obtain a contradiction;

We use $\oplus$ to denote the symmetrical difference (Xor) of X_1 and X_2. For example, if $X_1 = \{a_1\}$ and $X_2 = \{b_1\}$, then $X_1 \oplus X_2 = \{a_1, b_1\}$; if $X_1 = \{a_1\}$ and $X_2 = \{0\}$, then $X_1 \oplus X_2 = \{a_1\}$; if $X_1 = \{a_1\}$ and $X_2 = \{a_1, b_1\}$, then $X_1 \oplus X_2 = \{b_1\}$.

The function $\mathsf{UpdateMatrix}(B, \overline{X})$ updates the augmented matrix B according to the variable vector $\overline{X}$. If the i-th variable in $\overline{X}$ is 0, then the corresponding i-th column of B is set to a zero vector. As in [31], this method keeps solutions of the augmented matrix B unchanged. If $\overline{X}_i$ is not in the set $\{0, ?, *\}$, we check each row of B, if the value of the i-th column at the r-th row $B_{r,i}$ is 1, then we Xor $\overline{X}_i$ to the last element of the r-th row of B and set $B_{r,i}$ to 0.

Function UpdateMatrix(B, $\overline{X}$)

```
// Update the augmented matrix B according to the variable vector X̄
K ← the size of X̄;
for i ← 0 to K − 1 do
    t ← X̄[i] ;
    if t is 0 then
        Every element of the i-th column of B is set to 0.
    else if t is not '?' and t is not '*' then
        L ← the number of rows of B;
        for r ← 0 to L − 1 do
            if B[r, i] is 1 then
                B[r, i] ← 0;
                B[r, K − 1] ← B[r, K − 1] ⊕ t;
            end
        end
    end
end
```

The function $\mathsf{UpdateVector}(\overline{X}, \mathcal{N}, j, J)$ updates the j-th variable $\overline{X}_j$ with the value J, at the same time all constraints in $\mathcal{N}$ are maintained. As described in the beginning of this subsection, the function updates $\overline{X}_j$ with the value J by checking each constrint in $\mathcal{N}$ and returns true if succeeds or false if there is a contradiction. There are many subcases, ad described in the detailed algorithm. During the updating process, there may be contradictions. For example, if $\overline{X}_j = \{a\}$ and $J = \{a, b\}$

which means $J = a \oplus b$, there is a contradiction since $a \oplus b$ can never be a. If J is $\{0\}$ but the corresponding variable which is the sbox output of $\overline{X}_j$ is nonzero, or J is $\{0\}$ but the corresponding variable which is the sbox input of $\overline{X}_j$ is nonzero, there will be contradictions.

The function $\mathsf{ReducedRowEchelon}(B)$ transforming the $\iota \times \kappa$ matrix B into the reduced row echelon form by Gauss-Elimination algorithm. Note that every element in the first $\kappa - 1$ columns of B is in $\mathbb{F}_2$, while elements in the last column of B are represented by a set of symbols. Thus the Xor operation in the last column of B is the symmetrical difference operation. The readers can refer to [26] for the detailed algorithm of transforming a matrix into the reduced row echelon form.

The detailed algorithm for checking if a differential is impossible is described in Algorithm 1. In Algorithm 1, the variable vector $\overline{X}$ is first initialized according to the differential pair $(\Delta in, \Delta out)$ and the constraint array $\mathcal{N}$. Then the algorithm continue checks if there is a contradiction with a loop test until B and $\overline{X}$ is not updated any more. During the loop the algorithm first updates B according to $\overline{X}$ by the $\mathsf{UpdateMatrix}(B, \overline{X})$ function, and then transforms B into the reduced row echelon form by the $\mathsf{ReducedRowEchelon}(B)$ function to see if B has solutions. If B has no solutions, the algorithm obtains a contradiction and stops. Otherwise if there exists a solution for a variable from the reduced row echelon form, the index and the value of the variable is denoted as (j, J). The algorithm update the variable vector $\overline{X}$ with (j, J) by the $\mathsf{UpdateVector}(\overline{X}, \mathcal{N}, j, J)$ function , if the updating process return false, a contradiction is obtained and the algorithm stops, otherwise, the algorithm continues to run.

Function UpdateVector($\overline{X}$, $\mathcal{N}$, j, J)

`// Update the variable vector` $\overline{X}$ `according to the variable` (j, J) `where` J `is the value of the` j`-th variable in` $\overline{X}$`.` $\mathcal{N}$ `is the array of constraints.`

input : the variable vector $\overline{X}$, the constraint array $\mathcal{N}$, (j, J).
output: A boolean flag indicates if the update procedure success.

flag $\leftarrow$ **true**;
foreach $a \in \mathcal{N}$ **do**
  $k_0 \leftarrow a[0]$; $k_1 \leftarrow a[1]$;
  if j *is equal to* k_0 **then**
    if $J \oplus \overline{X}_{k_0}$ *is not 0* **then**
      flag $\leftarrow$ **false**; `// Ex.` $J = a \oplus b$ `but` $\overline{X}_{k_0} = a$`, a contradiction.`
      return *flag;*
    else if J *is* 0 ***and*** $\overline{X}_{k_0}$ *is ?* **then**
      if $\overline{X}[k_1]$ *is not 0* **then** flag $\leftarrow$ **false**;
        return *flag;*
      end
      $\overline{X}_{k_0} \leftarrow 0$; $\overline{X}_{k_1} \leftarrow 0$;
    else if J *is a nonzero value* **then**
      $\overline{X}_{k_0} \leftarrow J$; $\overline{X}_{k_1} \leftarrow *$;
    end
  else if j *is equal to* k_1 **then**
    if $J \oplus \overline{X}_{k_1}$ *is not 0* **then** flag $\leftarrow$ **false**
      return *flag;*
    else if $\overline{X}_{k_1}$ *is ?* ***and*** J *is 0* **then**
      $\overline{X}_{k_1} \leftarrow 0$; $\overline{X}_{k_0} \leftarrow 0$;
    end
  end
end
return *flag;*

Algorithm 1: The algorithm for checking an impossible differential

input : A differential pair $(\Delta in, \Delta out)$ and the system $\mathcal{S}$
output: A boolean flag indicates if $(\Delta in, \Delta out)$ is an impossible differential

```
B is the ι × κ augmented matrix of S;
X̄ is the κ − 1 dimension variable vector ;
N is the map of constraints of S;
flag←false;
index←true;
Initialize every variable in X̄ according to (Δin, Δout) and the constraints in N;
while index do
    UpdateMatrix (B, X̄)// Update B according to X̄ ;
    /* Transform B into the reduced-row-echelon form by Gauss-Jordan Elimination     */
    ReducedRowEchelon (B);
    if B has no solution then
        flag←true;
        break;
    else
        index ← false;
        count← 0 ;
        for i ← ι to 1 do
            v⃗ ← Row i of B ;
            if the sum of the first κ − 1 elements of v⃗ is 1 then
                j ← the index of the element 1 in v⃗;
                J ← the last element of v⃗; // the solution of the j-th variable in X̄
                /* update the variable vector X̄ with (j, J) and return true if there is
                   no contradiction and return false otherwise.                       */
                b ← UpdateVector (X̄, N, j, J);
                if b is false then
                    flag ← true;
                    return flag;
                else
                    index ← true;
                end
            end
        end
    end
end
return flag;
```

4.4 Complexity

For the $\iota \times \kappa$ matrix B and the $\kappa - 1$ dimension vector $\overline{X}$, the time complexity of the function UpdateMatrix is $\iota \cdot \kappa$, the time complexity of the function ReducedRowEchelon is $\iota^2 \cdot \kappa$, the time complexity of the function UpdateVector is a constant c. The while loop continues running $\kappa/2$ times since there at most $\kappa - 1$ values in $\overline{X}$ and in each loop either 2 variables are updated or there is a contradiction. Thus the total complexity of the algorithm is $\frac{c}{2} \cdot \iota^2\kappa^2$, where c is a small constant. The space complexity is dominated by storing the matrix B and is about $\iota \cdot \kappa$. The time complexity of the Wu-Wang method is $T \cdot \iota^2\kappa^2$ and this T is much larger than our c. The Wu-Wang method stores 3 matrices, thus its space complexity is at least triple of our method.

4.5 Comparison with Previous method

In [30], Wu and Wang proved that the $\mathcal{U}$-method and the UID-method are specific cases of the Wu-Wang method. They found that their method can find longer impossible differential for the MIBS

cipher than by $\mathcal{U}$-method and the UID-method. However, in the UID-method, for an impossible differential pair $((\Delta I_1, \ldots, \Delta I_n), (\Delta O_1, \ldots, \Delta O_n))$, the relationship between input variables and output variables are considered since UID-method uses symbols to denote values. For example, the UID-method considers the relation between ΔI_i and ΔO_j and checks if they are equal, however the $\mathcal{U}$-method and Wu-Wang method only use 0 and 1 to denote zero and non-zero values, which omit the relationship between input and output differentials.

Our improved method combines the advantages of the UID-method and Wu-Wang method. Every impossible differential found by the UID-method and Wu-Wang method can be found by our improved method. As Wu and Wang's method, impossible differentials found by our improved method must be correct if the algorithm is implemented correctly. Compare with Wu and Wang's method, our improved method is more complete. The symbol representation of a difference can represent more relationships between different difference values. Thus it can find more impossible differentials and the matrix B does not change with different values of $(\Delta in, \Delta out)$ in the beginning of the algorithm. While in the Wu-Wang method, to add linear relationships between nonzero values in $(\Delta in, \Delta out)$, the matrix B must change with different values of $(\Delta in, \Delta out)$. This will consume more time during the run of the algorithm.

The most time consuming part in the algorithm is the matrix operation. To check if the augmented matrix has any solutions, the Wu-Wang method needs to compute the rank of the matrix M and B. We show this step is not required since we can check the solvability of the system from the reduced row echelon form of the matrix B, as introduced in the preliminaries section. Thus our improvement largely reduces the search time of finding impossible differentials of a block cipher structure.

5 Applications and Experiment results

We implement the algorithm in java language and apply it to many block cipher structures, including Gen-CAST256 [25], Misty [24], Gen-Skipjack [29], Four-Cell [10], Gen-MARS [25], Gen-RC6 [25], SMS4 [27], MIBS [13], Camellia* [1,30], LBlock [32], E2 [14] and SNAKE [18]. We present the java code of this algorithm and complete impossible differential results in GitHub [22]. To reduce the space of this paper, we present some of the impossible differential results in Table 5. The file *Impossible Differential.txt* in [22] lists the complete impossible differential results for these block cipher structures. Most impossible differentials discovered by our algorithm are the same as the Wu-Wang method.

Block Cipher	UID [23]	Wu-Wang [30]	This paper
Gen-Skipjack	16: $(0,0,0,a) \nrightarrow_{16} (b,0,0,b)$	-	same as UID
Gen-CAST256	19: $(0,0,0,a) \nrightarrow_{19} (a,0,0,0)$	-	same as UID
Four-Cell	18: $(a,0,0,0) \nrightarrow_{18} (b,b,0,0)$	-	same as UID
Gen-MARS	11: $(0,0,0,a) \nrightarrow_{11} (a,0,0,0)$	-	same as UID
Gen-RC6	9: $(0,0,a,0) \nrightarrow_{9} (0,a,0,0)$ $(a,0,0,0) \nrightarrow_{9} (0,0,0,a)$	-	same as UID
SMS4	11: $(0,0,0,a) \nrightarrow_{11} (a,0,0,0)$	-	same as UID
Misty	-	-	$4 : (0,a) \nrightarrow_{4} (b,b)$
SNAKE	-	-	$11 : (0,0,0,0,0,0,a,0) \nrightarrow_{11} (0,0,b,0,0,0,0,0)$
Camellia*	-	8-round, 4 IDs	same as Wu-Wang
MIBS	-	8-round, 6 IDs	8-round, 10 IDs
LBlock	-	14-round, 80 IDs	same as Wu-Wang
E2	-	6-round, 56 IDs	same as Wu-Wang

Table 2. Summary of Impossible differentials (IDs) of some well-known block ciphers structures found by different methods

Moreover, for the 8-round MIBS, we find new 4 impossible differentials, which are not found by the Wu-Wang method since these new 4 impossible differentials are not simple truncated impossible differentials. MIBS is a 16-subblock Feistel structure with substitution and permutation (SP) round function. In the SP round function, the 8 subblock is first substituted by 8 sboxes, then a 8×8

matrix is applied as the permutation. The permutation matrix P is

$$\begin{pmatrix} 1&1&0&1&1&0&1&1 \\ 0&1&1&1&1&1&1&0 \\ 1&1&1&0&1&1&0&1 \\ 0&1&1&1&0&0&1&1 \\ 1&0&1&1&1&0&0&1 \\ 1&1&0&1&1&1&0&0 \\ 1&1&1&0&0&1&1&0 \\ 1&0&1&1&0&1&1&1 \end{pmatrix}.$$

There are total 10 impossible differentials are found for 8-round MIBS by our improved algorithm. The new four 8-round impossible differential found are listed in Table 3.

No.	Δin	Δout	Reference
1	$(0,0,0,0,0,0,0,0;0,0,0,0,0,0,a,0,0)$	$(b,0,0,0,0,0,0,b;0,0,0,0,0,0,0,0)$	This paper
2	$(0,0,0,0,0,0,0,0;0,0,0,0,0,0,a,0,0)$	$(0,0,0,0,b,0,0,b;0,0,0,0,0,0,0,0)$	
3	$(0,0,0,0,0,0,0,0;a,0,0,0,0,0,0,a)$	$(0,0,0,0,0,b,0,0;0,0,0,0,0,0,0,0)$	
4	$(0,0,0,0,0,0,0,0;0,0,0,0,a,0,0,a)$	$(0,0,0,0,0,b,0,0;0,0,0,0,0,0,0,0)$	
5	$(0,0,0,0,0,0,0,0;0,0,a,0,0,0,0,0)$	$(0,0,0,0,b,0,0,0;0,0,0,0,0,0,0,0)$	[30]
6	$(0,0,0,0,0,0,0,0;0,0,a,0,0,0,0,0)$	$(0,0,0,0,0,0,0,b;0,0,0,0,0,0,0,0)$	
7	$(0,0,0,0,0,0,0,0;0,0,0,0,a,0,0,0)$	$(0,0,b,0,0,0,0,0;0,0,0,0,0,0,0,0)$	
8	$(0,0,0,0,0,0,0,0;0,0,0,0,a,0,0,0)$	$(0,0,0,0,0,0,b,0;0,0,0,0,0,0,0,0)$	
9	$(0,0,0,0,0,0,0,0;0,0,0,0,0,0,a,0)$	$(0,0,0,0,b,0,0,0;0,0,0,0,0,0,0,0)$	
10	$(0,0,0,0,0,0,0,0;0,0,0,0,0,0,0,a)$	$(0,0,b,0,0,0,0,0;0,0,0,0,0,0,0,0)$	

Table 3. Impossible differentials for 8-round MIBS. There are 4 new found impossible differentials. a and b are nonzero values and a and b can have the same value.

Compare with Wu and Wang's algorithm, this improvement is more general since it not only finds more impossible differentials for a block cipher structures, but also has better efficiency. The results for MIBS are obtained on a 2.66 GHz processor with MAGMA package in a few hours by Wu and Wang's algorithm [30]. However, our results for MIBS are obtained on a 2.20 GHz processor in Java language in less than 10 seconds. Thus, the algorithm presented in this paper is more efficient than Wu and Wang's algorithm.

6 Conclusion

In this paper we improve Wu and Wang's algorithm for finding impossible differentials of block cipher structures. The improved method is more general than Wu and Wang's method that it can find more impossible differentials with less time. We apply this method to many block cipher structures. The experiment results show that this improvement can largely reduce the search time for the impossible differentials of a block cipher. Since there are known relationships between impossible differential, integral and zero correlation linear cryptanalysis [7,5,28]. This method can be used as a cryptanalytic tool to evaluate the security of a block cipher against these kinds of cryptanalysis.

References

1. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., and Tokita, T. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In *SAC 2000* (2011), vol. LNCS 2012, Springer-Verlag, pp. 39–56.
2. Berger, T., and Minier, M. Some results using the matrix methods on impossible, integral and zero-correlation distinguishers for Feistel-Like ciphers. In *Progress in Cryptology – INDOCRYPT 2015* (2015), vol. LNCS 9462, pp. 180–197.

3. Berger, T., Minier, M., and Thomas, G. Extended generalized Feistel networks using matrix representation. In *Selected Areas in Cryptography – SAC 2013* (2014), vol. LNCS 8282, pp. 289–305.
4. Biham, E., Biryukov, A., and Shamir, A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible diffrentials. In *Advances in Cryptology - EUROCRYPT'99* (1999), vol. LNCS 2595, Springer-Verlag, pp. 12–23.
5. Blondeau, C., Bogdanov, A., and Wang, M. On the (in)equivalence of impossible differential and zero correlation distinguishers for Feistel and Skipjack-type ciphers. In *ACNS 2014* (2014), vol. LNCS 8479, Springer-Verlag, pp. 271–288.
6. Blondeau, C., and Minier, M. Analysis of Impossible, Integral and Zero-Correlation Attacks on Type-II Generalized Feistel Networks Using the Matrix Method. In *FSE 2015* (2015), vol. LNCS 9054, Springer-Verlag, pp. 92–113.
7. Bogdanov, A., Knudsen, L., Leander, G., Standaert, F., Steinberger, J., and Tischhauser, E. Key-Alternating Ciphers in a Provable Setting: Encryption Using a Small Number of Public Permutations. In *ASIACRYPT 2012* (2012), vol. LNCS 7237, Springer Verlag, pp. 45–62.
8. Boura, C., Naya-Plasencia, M., and Suder, V. Scrutinizing and Improving Impossible Differential Attacks: Applications to CLEFIA,Camellia, LBlock and Simon. In *ASIACRYPT 2014, PART I* (2014), vol. LNCS 8873, Springer-Verlag, pp. 179–199.
9. Cheng, L., Sun, B., and Li, C. Revised cryptanalysis for sms4. *Science China Information Sciences 60*, 12 (2017), 122101.
10. Choy, J., Chew, G., Khoo, K., and Yap, H. Cryptographic properties and application of a generalized unbalanced Feistel network structure. In *Proc of ACISP'2009* (2009), vol. LNCS 5594, Spring-Verlag, pp. 73–89.
11. Dai, Y., and Chen, S. Cryptanalysis of full PRIDE block cipher. *Science China Information Sciences 60*, 5 (2017), 052108.
12. Ding, Y., Zhao, J., Li, L., and Yu, H. Impossible differential analysis on round-reduced PRINCE. *J. Inf. Sci. Eng. 33*, 4 (2017), 1041–1053.
13. Izadi, I., Sadeghiyan, B., Sadeghian, S., and Khanooki, H. MIBS: A new lightweight block cipher. In *CANS 2009* (2009), vol. LNCS 5888, Springer-Verlag, pp. 334–348.
14. Kanda, M., Moriai, S., Aoki, K., Ueda, H., Takashima, Y., Ohta, K., and Matsumoto, T. E2-A new 128 bit block cipher. *IEICE Transactions Fundamentals - Special Section on Cryptography and Information Security E83-A*, 1 (2000), 48–59.
15. Kim, J., Hong, S., and Lim, J. Impossible differential cryptanalysis using matrix method. *Discrete Mathematics 310* (2010), 988–1002.
16. Kim, J., Hong, S., Sung, J., Lee, S., and Lim, J. Impossible differential cryptanalysis for block cipher structures. In *INDOCRYPT 2003* (2003), vol. LNCS 2904, Springer-Verlag, pp. 82–96.
17. Knudsen, L. R. DEAL-A 128-bit block cipher. Technical Report 151. Tech. rep., Department of Informatics, University of Bergen, 1998.
18. Lee, C., and Cha, Y. The block cipher: SNAKE with provable resistance against DC and LC attacks. In *JW-ISC 1997* (1997), pp. 3–17.
19. Lin, T., Lai, X., Xue, W., and Huang, G. Discussion on the theoretical results of white-box cryptography. *Science China Information Sciences 59*, 11 (2016), 1–11.
20. Liu, G., and Jin, C. Algebraic techniques in slender-set differential cryptanalysis of PRESENT-like cipher. *Science China Information Sciences 59*, 9 (2016), 099104.
21. Liu, G., Jin, C., and Kong, Z. Key recovery attack for PRESENT using slender-set linear cryptanalysis. *Science China 59*, 3 (2016), 1–14.
22. Luo, Y. Source codes and results for finding impossible differentials for block cipher structures. https://github.com/ianroo/impossibledifferential.
23. Luo, Y., Lai, X., Wu, Z., and Gong, G. A unified method for finding impossible differentials of block cipher structures. *Information Sciences 263* (2014), 211–220.
24. M.Matsui. New block encryption algorithm MISTY. In *FSE 1997* (1997), vol. LNCS 1267, Springer-Verlag, pp. 54–68.
25. Moriai, S., and Vaudenay, S. On the pseudorandomness of top-level schemes of block ciphers. In *Advances in Cryptology - ASIACRYPT'00* (2000), vol. LNCS 1976, Springer-Verlag, pp. 289–302.
26. RosettacodeOrg. How to compute the reduced row echelon form of a matrix *http://rosettacode.org/wiki/Reduced_row_echelon_form*.
27. SMS4. Specication of SMS4, block cipher for WLAN products SMS4 (in Chinese). Avaiable at: http://www.oscca.gov.cn/UpFile/2006210164231979 90.pdf.
28. Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., Alkhzaimi, H., and Li, C. Links among impossible differential, integral and zero correlation linear cryptanalysis. In *EUROCRYPT 2015, Part I* (2015), vol. LNCS 9215, Springer-Verlag, pp. 95–115.

29. Sung, J., Lee, S., Lim, J., Hong, S., and Park, S. Provable security for the Skipjack-like structure against differential cryptanalysis and linear cryptanalysis. In *Advances in Cryptology-ASIACRYPT'00* (2000), vol. LNCS 1976, Springer-Verlag, pp. 274–288.
30. Wu, S., and Wang, M. Automatic search of truncated impossible differentials for word-oriented block ciphers. In *http://eprint.iacr.org/2012/214.pdf*.
31. Wu, S., and Wang, M. Automatic search of truncated impossible differentials for word-oriented block ciphers. In *INDOCRYPT 2012* (2012), vol. LNCS 7668, pp. 283–302.
32. Wu, W., and Zhang, L. LBlock: A lightweight block cipher. In *ACNS 2011* (2011), vol. LNCS 6715, Springer-Verlag, pp. 227–344.
33. Wu, W. L., Zhang, L., and Yu, X. L. The DBlock family of block ciphers. *Science China Information Sciences 58*, 3 (2015), 1–14.
34. Yap, H. Impossible differential characteristics of extended Feistel networks with provable security against differential cryptanalysis. In *Proc of SecTech 2008* (2009), vol. CCIS 29, pp. 103–121.
35. Zhang, W. T., Bao, Z. Z., Lin, D. D., Rijmen, V., Yang, B. H., and Verbauwhede, I. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences 58*, 12 (2015), 1–15.
36. Zhao, R., Zhang, R., Li, Y., and Wu, B. Construction of MDS block diffusion matrices for block ciphers and hash functions. *Science China Information Sciences 59*, 9 (2016), 1–3.
37. ZONG, R., Dong, X., and Wang, X. Impossible Differential Attack on Simpira v2. Science China Information Sciences,DOI:10.1007/s11432-016-9075-6, 2017.