Research Article

New Search Method for Sbox-Related Impossible Differentials

Yi Zhang,1 Xuan Shen,2 Guoqiang Liu,1,3,4 and Chao Li1,3

1College of Liberal Arts and Sciences, National University of Defense Technology, Changsha 410073, China
2College of Information and Communication, National University of Defense Technology, Wuhan 430010, China
3Hunan Engineering Research Center of Commercial Cryptography Theory and Technology Innovation, Changsha 410073, China
4State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Correspondence should be addressed to Chao Li; lichao_nudt@sina.com

Received 5 May 2022; Revised 7 July 2022; Accepted 27 July 2022; Published 5 September 2022

Academic Editor: Angel M. Del Rey

The impossible differential attack is one of the most fundamental tools of cryptanalysis and has been successfully applied to a large variety of block ciphers. In a typical impossible differential attack, the foundation and first step is to construct an impossible differential. Nowadays, two kinds of most commonly used approaches in impossible differential construction are the matrix-based and tool-aided automatic search methods. In this paper, we proposed a new search method combining an early-abort strategy with the guess-and-determine technique to find longer impossible differentials. Compared with the previous matrix-based search methods, ours has taken the details of Sbox into consideration, while compared with the tool-aided methods, ours is independent of a third-party solver and is applicable to ciphers with large (≥8 bits) Sboxes, which could be tough work for tool-aided methods to cover. Therefore, more accurate results could be obtained. To prove the effectiveness of our method, it is applied to CSA and FOX64. For CSA, 23/24/25-round impossible differentials were found for CSA-BC. This improved the longest impossible differential distinguisher so far by 1/2/3 rounds. And 25-round key recovery attacks are performed against CSA-BC and CSA, which are the longest impossible differential attack by now. For FOX64, we proved that an impossible differential for its round function always implies an impossible differential for 4-round FOX64, and new types of impossible differentials were able to be found. Our method has its own advantage in dealing with large Sboxes, and it may be a possible direction to search for Sbox-related impossible differentials.

1. Introduction

Block ciphers play a fundamental role in symmetric-key cryptography. Apart from encryption, it could also be used as primitives in stream ciphers, hash functions, authenticated encryption schemes, and so on. The confidence in the security of a block cipher is largely derived from its resistance against known attacks, among which the differential attack and its variants might be one of the most well-known instances. The differential attack has been successfully applied to many block ciphers, stream ciphers, and hash functions, and numerous variants have been derived, including impossible differential cryptanalysis, truncated differential cryptanalysis, higher order differential cryptanalysis, multiple differential cryptanalysis, and differential-linear cryptanalysis. Among them, the impossible differential attack is one of the most basic tools for block cipher analysis. The impossible differential attack was proposed by Knudsen [1] and Biham et al. [2] independently and has been applied to large numbers of block ciphers successfully, such as AES, Skipjack, and HIGHT [2–4].

As a variant of differential attack, impossible differential attack exploits differentials, which occur with probability zero. Usually, two steps are taken to recover keys in a typical impossible differential attack. The first step is to detect a differential for the reduced-round target block cipher with probability zero. The second step is to guess some key materials and eliminate those that satisfy a probability-zero differential distinguisher. During the attack procedure, constructing impossible differentials in the first step is the key point and foundation of impossible differential attacks.
1.1. Construction of Impossible Differentials. An impossible differential is usually constructed by the miss-in-the-middle strategy. First, it propagates the input difference forward and the output difference backward with probability 1 for some rounds. Then, if a contradiction is detected in the middle state, the differential is supposed to be impossible.

With ad hoc approaches for a specific cipher being studied extensively, generic search models for impossible differentials have been developed. In [5], Kim et al. proposed a matrix-based search method called ‘U’-method. They first convert the differential propagation for one round encryption or decryption by a matrix, which identifies the relationship between the input and output blocks. Then, a matrix operation is defined and applied to describe round iteration. If contradictions are found in the middle state matrix, an impossible differential is detected and could further be specified. This method is extended to the UID [6] and Wu and Wang [7] method with a more detailed description of the structure and contradiction afterward. When applied to a specific block cipher of structure, this method could utilize the properties of those structures, and better results are acquired [8–10]. Those matrix-based methods are easy to implement and require little computational and space resources. However, information on the nonlinear component is hardly used except that the Sboxes are bijective.

A generic and systematic method to take detailed information on Sboxes into consideration in impossible differential construction has not appeared until the automatic analytical model was introduced to cryptanalysis. In 2011, Mouha et al. [11] used the mixed-integer linear programming (MILP) model to search the security bounds against differential and linear attacks. Then, these kinds of automatic tools are widely applied to integral, meet-in-the-middle, rotational-XOR differential attacks and so on [12–14]. In EUROCRYPT 2017, an MILP-based automatic search model was developed [15] for impossible differential attack. In this method, the differential properties of linear and nonlinear components are expressed as linear inequalities, and the differential search process is modeled as an inequality system and then fed to a third-party solver. If some error codes indicating the system is infeasible are returned, then the differential is expected to be impossible. In ASIACRYPT 2020 [16], another automatic search model was put forward to search for impossible differentials by describing the propagation of values instead of differences. With those automatic tools, the precision of analytical results is improved. However, when the size of the Sbox is large (e.g., ≥8 bits), modeling the properties of the Sbox usually involves large numbers of inequalities, which is a tough task for the third-party solvers to handle.

1.2. Our Contributions. In this paper, we proposed an Sbox-related impossible differential search method and further improved it by introducing the guess-and-determine and the early-abort technique to reduce the complexity. To demonstrate the effectiveness of our method, we applied it to CSA-BC and FOX64 whose Sboxes are both of 8-bit size. Our main contributions are as follows:

(i) Based on the guess-and-determine strategy and early-abort technique, a new search method for Sbox-related impossible differentials was developed. Compared with previous matrix-based search methods, ours could exploit more detailed information on the nonlinear primitives. At the same time, our search method is independent of third-party solvers and possesses high performance in dealing with large size Sboxes.

(ii) For CSA-BC, 23/24/25-round impossible differentials were found with our search method in less than 5 seconds, which improved previous works by 1/2/3 rounds. Then, we mounted a 25-round impossible differential attack against CSA-BC using the impossible differentials found by our method. Finally, the key recovery attack is extended to 25-round CSA. And those key recovery attacks are the longest impossible differential attacks against CSA-BC and CSA by now. The results are listed in Table 1.

(iii) For FOX64, the relationship between 4-round impossible differentials and its round function is discovered. Then, a framework is derived to verdict whether a 4-round differential

\[ (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8) \rightarrow (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8) \]  

is possible or not, which allows more flexibility in differential patterns compared with previous works. And for the first time, detailed information on Sboxes is taken into consideration in the construction of impossible differentials for FOX64. As a result, new types of impossible differentials were able to be identified. The results are listed in Table 2.

The rest of this paper is organized as follows. In Section 2, we will introduce the basics. Method to construct impossible differential is introduced in Section 3. The results of CSA-BC and FOX64 are presented in Sections 4 and 5, respectively. Section 6 concludes this paper.

2. Preliminaries

This section will describe the impossible differential and some techniques to reduce complexity in cryptanalysis. Before that, we will first introduce some notations used throughout this paper.

Denote by \( \mathbb{F}_q \) the finite fields with \( q \) elements, and \( \mathbb{F}_q^* \) contains all nonzero elements of \( \mathbb{F}_q \). Numbers over \( \mathbb{F}_2 \) are expressed in hexadecimal, and the prefix ‘0x’ is omitted. The number of elements in set \( A \) is denoted by \( |A| \). For simplicity, consecutive variables \( v_i, v_{i+1}, \ldots, v_j \) are denoted as \( v_{i:j} \).

2.1. Impossible Differential. The difference for \( x, y \in \mathbb{F}_2^q \) is defined as \( u = x \oplus y \). A differential \( (u, v) \) for \( F^{(0)} \) is that the input difference of \( F^{(0)} \) is \( u \), and its output difference is \( v \). When there does not exist any plaintext that satisfies the differential, that is,
2.2. Guess-and-Determine Technique. The guess-and-determine technique is a commonly used technique to decrease the computational workload in cryptanalysis, and it also has wide application in stream cipher cryptanalysis [19–21]. The guess-and-determine technique exploits the relationship among internal values. It guesses certain bits and makes use of the relationship to determine the remaining bits, thus decreasing the complexity of trying the remaining bits. A typical procedure of the guess-and-determine technique is to derive the relationship first and then exhaustively guess over a relatively small proportion of state to recover the whole state. Finally, the attacker checks the correctness by comparing it with known conditions. Assume that the exploited internal relationship $f$ maps $t$-bit ($t < n$) state $s_i$ to other $(n-t)$-bit state $s_{n-t}$; say

$$f(x_0, x_1, \ldots, x_{t-1}, x_t, \ldots, x_{n-1})$$

then this differential is said to be impossible. To simplify the probability evaluation of differentials, differential characteristic is proposed.

An $r$-round differential characteristic for iterated block cipher is defined as $(u_0, u_1, \ldots, u_r)$ that the input difference is $u_0$ and the output difference after round $i$ is $u_i$. Then, the probability of the differential is the sum of all its differential characteristics. A differential is impossible, implying that all characteristics are impossible. Otherwise, there exists at least one characteristic whose probability is greater than zero. That is, denote by $Pr(u_0, u_1, \ldots, u_r)$ the probability for differential characteristic $(u_0, u_1, \ldots, u_r)$, and then an $r$-round differential $(u, v)$ is impossible if for all $u_i \in \mathbb{F}_2^n$, $1 \leq i \leq r - 1$,

$$\sum_{u_0, u_1, \ldots, u_r \in \mathbb{F}_2^n, 1 \leq i \leq r-1} Pr(u_0, u_1, \ldots, u_r) = 0. \quad (3)$$

Otherwise, the differential is possible, and there exists a set of $u_i$ such that

$$Pr(u_0, u_1, \ldots, u_r) > 0. \quad (4)$$

### Table 1: Impossible differential attack against CSA-BC and CSA.

| Cipher         | $r_{\text{Key}}$ | $r_{\text{Dist}}$ | Data complexity | Time complexity | Reference |
|----------------|-------------------|-------------------|-----------------|-----------------|-----------|
| CSA-BC         | 21                | 20                | $2^{44.5}$      | $2^{22.7}$      | [17]      |
| CSA-BC*        | 22                | 22                | \_              | \_              | [18]      |
| CSA-BC         | 25                | 23/24/25          | $2^{58.5}$      | $2^{61.5}$      | Section 4 |
| CSA            | 21                | 20                | $2^{47.5}$      | $2^{21.7}$      | [17]      |
| CSA            | 25                | 24                | $2^{58.5}$      | $2^{61.5}$      | Section 4 |

$r_{\text{Key}}/r_{\text{Dist}}$ = length of key recovery attack/distinguisher. *The key recovery attack may be invalid since not enough data could be collected.

### Table 2: 4-round impossible differentials for FOX64.

| Impossible differential pattern | Constraints | Reference |
|--------------------------------|-------------|-----------|
| $(0, 0, 0, u) \mapsto (v_1, v_2, v_3, v_1)$ | $u, v_1 \neq 0$ | [33]      |
| $r^{-1}(u) \mapsto \sigma(v)$ | $u = v$ | [34]      |
| $\sigma(v) \mapsto i_0(v)$ | $wt(u) + wt(u \oplus v) \leq 4$ | [35]      |
| $(0, 0, u_1, u_2) \mapsto (v_1, v_2, v_3, u_1)$ | $u_1, v_1 \neq 0$ | Section 5 |

Note that the differences in the two branches of FOX64 are equal; for simplicity, only one branch is shown in this table.

$$\# \{ x \in \mathbb{F}_2^n | f^{(r)}(x) \oplus f^{(r)}(x \oplus u) = v \} = 0, \quad (2)$$

2.3. Early-Abort Technique. The early-abort technique is noticed as early as in the differential attack against DES [22]; it has numerous successful applications in key recovery attacks against block ciphers and public-key encryption schemes designing [23, 24]. Instead of guessing all the required unknown bits at a time, it partially checks whether a candidate pair could produce the expected difference by guessing only a small fraction of keys. Since some useless pairs are discarded before the next guess, the computational workload could be decreased consequently. When this idea is applied to the search for impossible differential, according to (3) and (4), if one possible characteristic is detected, then this differential must be possible, and the check process for other characteristics could be terminated. In addition, when testing whether a single characteristic is possible or not, if one condition is found to be dissatisfied, this characteristic is impossible, and the cost of computing other variables would be saved.

3. Construction of Impossible Differentials

In this section, we will propose a generic framework to search for Sbox-related impossible differentials. Then the guess-and-determine and the early-abort techniques are introduced to improve the performance of the search algorithm. Our method puts no constraints on the structure of block ciphers, and it could be applied to SPN, Feistel, Lai-Massey, or other structures of block ciphers.

In order to explain our techniques better, in the following, we will use the small-scale AES proposed by Cid et al.
as an example. This variant inherits the design features of AES and provides a suitable instance for different cryptanalytic methods. The small-scale version [25] parameterized by (2, 2, 2, 8), also called miniAES for short in this section, has 2 rounds and a block size of 32 bits, where the data block is viewed as an array of \( 2 \times 2 \) words of 8 bits.

The encryption of miniAES is similar to AES. In the SP type round functions, an 8-bit Sbox is applied to each word first, which is the SB operation. Then the SR operation shifts the second row for 1 word. Finally, the MC operation multiplies a matrix over \( \mathbb{F}_{2^2} \) and the state array. The matrix is specified as

\[
\begin{bmatrix}
3 & 2 \\
2 & 3
\end{bmatrix} \in \mathbb{F}_{2^2}^{2\times2}.
\]  

(6)

Note that the MC operation in the last round is omitted. Since the impossible differential properties are regarded as independent of round key addition, the AK operation is omitted here.

In the following, we will study whether the 2-round differential

\[
(u_0, 0, 0, u_1) \rightarrow (v_0, 0, 0, v_1)
\]  

(7)

is possible or not for miniAES, where \( u_i, v_i \in \mathbb{F}_{2^2}^* \). The differential propagation is presented in Figure 1. Since the output difference of Sbox is hard to predict, new variables \( u_2, u_3, v_2, v_3 \) are introduced to show the propagation of Sboxes. When the details of Sboxes are taken into consideration, the impossible differential studied may not be in the truncated form. It is impossible for some specific instances, and its value is closely related to the choice of Sbox; this is what we called Sbox-related impossible differentials in our paper. It should also be pointed out that the input and output difference pattern is chosen empirically or by analyzing the structure of the target cipher.

### 3.1. A Straightforward Search Method for Impossible Differentials

To verify whether a differential is possible or not, according to the definition, that is, (8), all differential characteristics are impossible if the differential is impossible. Therefore, all characteristics should be checked. The conditions used in the check procedure could be divided into two parts. The first is that the differences in the middle state from both encryption and decryption should match where a linear equation system is built. For miniAES, this equation system is

\[
\begin{align*}
v_2 &= 3u_2 \oplus 2u_3, \\
v_3 &= 2u_2 \oplus 3u_3.
\end{align*}
\]  

(8)

The second is that new variables are introduced for active Sboxes; therefore, they should be in accordance with the propagation rules of Sboxes; that is to say, the corresponding entries in DDT are nonzero. Denote the entry of DDT as

\[
DDT[\alpha, \beta] = |\{ x \in \mathbb{F}_{2^2} : S(x) \oplus S(x \oplus \alpha) = \beta \}|. 
\]  

(9)

For miniAES, it is required that

\[
DDT[u_0, u_2] > 0, DDT[u_1, u_3] > 0, DDT[v_0, v_2] > 0, DDT[v_1, v_3] > 0. 
\]  

(10)

In the search for impossible differentials, all variables are enumerated first, where each set of \( u_i, v_i \) corresponds to a differential characteristic. If both (8) and (10) are satisfied, then this characteristic is possible. This definition-based method is shown in Algorithm 1.

**Remark 1.** Algorithm 1 provides an efficient way to take the DDT of Sbox into consideration. In our method, the information on Sbox is exploited by checking whether the differential propagation is in accordance with the DDT as in equation (10), while for the MILP-based search method, modeling a single 8-bit Sbox involves thousands of inequalities. And in our method, the DDT is accessed by table lookup, which is more effective. Moreover, when the size of Sboxes increases, the computation complexity of table lookup would still remain effective.

**Remark 2.** In the matrix-based method and its derivatives, the propagation rule of \( n \)-bit Sbox is that a nonzero input difference \( \delta^* \in \mathbb{F}_{2^n} \) will propagate to a nonzero output difference \( \Delta^* \in \mathbb{F}_{2^n} \), where \( \Delta^* \) could be any value over \( \mathbb{F}_{2^n}^* \). However, the exact value of the output difference usually is not considered. Therefore, the middle state conditions are built, and the attackers detect contradictions where the difference from one side is nonzero while it is zero from the other side of the equations. In fact, for an Sbox over a finite field with characteristic 2, the proportion of possible output difference \( \Delta^* \) for a given input difference is less than 50%, and treating \( \Delta^* \) as any nonzero value might cause inaccuracies consequently. In our method, only the output difference with \( DDT[\delta^*, \Delta^*] > 0 \) is taken into consideration, resulting in a more accurate description of the differential propagation. Furthermore, for the middle state conditions, it is checked whether the exact value of the differences from both sides matches.

Though the definition-based search method could take more detailed differential properties into consideration, checking all characteristics will require excessive time and memory sources. Therefore, in the following, the guess-and-determine and the early-abort technique will be introduced to improve its performance in the enumerating process and the checking process.

### 3.2. Improvement with Guess-and-Determine Technique

The guess-and-determine technique exploits the relationship among internal values to reduce the complexity of enumerating all variables. For miniAES, there are eight 8-bit variables \( u_0, u_1, u_2, u_3, v_0, v_1, v_2, v_3 \), that is, \( 8 \times 8 = 64 \) bits in total. Therefore, \( 2^{64} \) guesses would be needed to search for this type of impossible differentials. However, \( v_2 \) and \( v_3 \) could be expressed as functions of other variables as in (8). By exploring this relationship, instead of guessing all the 64 bits and then checking (8) and (10), the search could be performed by guessing \( u_0, u_1, u_2, u_3, v_0, v_1 \) first, then
computing $v_2$ and $v_3$ by (8), and finally checking by (10). By exploiting the linear relationship among variables, the guesses of $v_2$ and $v_3$ could be saved. Furthermore, the resultant state will match the middle state since they are derived from it. Therefore, the checking process will also be reduced.

The above process could be generalized as follows. Assume that the input difference is encrypted by $r_1$ rounds, and the output difference is decrypted by $r_2$ rounds. To precisely describe the differential propagation, a variable is introduced if its difference is not determined. Usually, new variables are introduced for the output of an active Sbox or the input of the inverse of an active Sbox. Assuming that there are $l$-bit variables in total introduced for the $(r_1 + r_2)$-round differentials, to check whether there exist $(r_1 + r_2)$-round impossible differentials, enumeration of all the $2^l$ possible states is required. Using the guess-and-determine technique, when checking the possibility for every characteristic, the middle state computed from encryption and decryption direction shall match, resulting in a linear equation system. By Gaussian elimination, the controlled variables could be expressed by free variables. In the example of miniAES, the equation system is (3), and the controlled variables are $v_2$ and $v_3$, which are expressed by $u_3$ and $u_4$. If $n$-bit ($n < l$) controlled variables could be derived by the equation system, then only $2^{n-l}$ states are required to try through instead of enumerating all $2^l$ states. This decreases the complexity by a factor of $2^n$. Taking all those conditions into consideration, an algorithm is derived by combining the definition-based method and the guess-and-determine technique, which could reduce the complexity of enumerating and is shown in Algorithm 2.

The difference lies in that, in Algorithm 1, all variables are guessed as in Line 3, and the conditions are checked for Sbox and middle state in Line 4, while in Algorithm 2, the equation system is first derived in Line 2–4. Then only the free variables are enumerated in Line 5, and the controlled variables are computed accordingly in Line 6. Finally, only the conditions of the Sbox need to be checked in Line 7.

The time-consuming step for the search process is the loop in Line 5. Taking CSA-BC as an example, to search for $r$-round ($r = r_1 + r_2$ and $r \geq 20$) impossible differentials, $(r - 12)$-bit variables will be introduced. Among them, there are 8 controlled 8-bit variables, and the remaining $(r - 20)$-bit variables are free. Therefore, the complexity for searching $r$-round impossible differential is $O(2^8(r-20))$. That is to say, the complexity of searching for 25-round and 26-round distinguishers is $2^{40}$ and $2^{48}$, respectively, where hours or days are needed for personal computers to cover the workload; thus, further improvement is needed.

3.3. Improvement with Early-Abort Technique. Back to the example of miniAES, according to (3) and (4), while all differential characteristics are required to be impossible for an impossible differential, it is enough to deny it if there exists one possible, meaning that the checking process could be terminated when a possible differential is detected. That is to say, when a set of $(u_0, u_1, u_2, u_3, v_0, v_1)$ is detected to satisfy the conditions of Sbox, meaning the corresponding differential characteristic in Figure 1 is possible, then the differential $(u_0, 0, 0, u_1) \rightarrow (v_0, 0, 0, v_1)$ is possible, and there is no need to guess other $u_2$ and $u_3$ candidates. Moreover, there is no need to compute all controlled variables at a time. Instead, we compute one and check immediately. If the conditions are not satisfied, then this characteristic would be impossible, and we continue with another. For miniAES, the checking conditions for the middle state are guaranteed by the guess-and-determine technique, and the checking conditions for Sboxes are shown in (10). After guessing $u_0$ and $u_1$, whether $DDT[u_0, u_1] > 0$ is checked immediately. If this is not satisfied, this characteristic is impossible, and there is no need to guess or compute other variables. Similarly, computing $v_2$ is followed by checking whether $DDT[v_0, v_2] > 0$ immediately. Performing the check procedure in advance would save much effort in guessing and computing those useless candidates. Based on those, an improved search algorithm combining the early-abort technique is proposed as shown in Algorithm 3, where the free variables are distinguished by whether they are involved in the first or last round.

The improvement by the early-abort technique is that there is no need to compute all characteristic or controlled variables. In one trial, if some checking conditions are not satisfied (Line 7 and Line 15), indicating that this characteristic is impossible, then this loop is terminated. If one characteristic is found to be possible (Line 10), that is, all controlled variables are computed and passed the check procedure, then this differential is possible, and the loop for testing other characteristics is stopped. Thus, the computation and determination could be saved. The application of this algorithm will be presented in the next sections.

To prove the validity of our algorithm, we performed comparative experiments on a personal computer. The device is equipped with an Intel Core i7-6500U processor, and the running time for finding an impossible differential for CSA-BC is shown in Table 3. The results show that, for Algorithm 2, time consumed grows sharply with the increase of rounds, while the impossible differential for 22–26 rounds could be acquired in less than 5 seconds with Algorithm 3. Moreover, it is interesting to notice that the running time decreases when the length of differentials increases from 25
to 26. This is attributed to the effect of the early-abort technique. The proportion of possible characteristics grows exponentially with the increase of length; therefore, the search process would be terminated in advance, and much effort is saved thanks to the early-abort technique.

4. Constructing Impossible Differentials for CSA-BC

In this section, we will first give a brief introduction of CSA-BC and then present the results of impossible differentials constructed by our method. The impact of Sbox on impossible differentials will also be discussed. Finally, key recovery attacks will be mounted using the impossible differentials found.

4.1. Description of CSA-BC. Digital Video Broadcast (DVB) is a series of standards for digital television. It is widely applied in most countries and regions in the world. The Digital Video Broadcast Common Scrambling Algorithm (DVB-CSA, or CSA for short) is mainly specified to guarantee the secure transmitting of MPEG-2 signal in Pay-TV and was adopted by the DVB consortium in 1994. The encryption process of CSA can be divided into two parts, the block cipher part (CSA-BC) and the stream cipher part (CSA-SC).

CSA-BC is a 64-bit block cipher with a 64-bit key size. The state could be regarded as an 8-byte vector. And the plaintext is updated by the round function \( F \) for 56 rounds.

The round function \( F \) is depicted in Figure 2 and works as follows. Denote its input and output as \( (x_0, x_1, \ldots, x_7) \) and \( (y_0, y_1, \ldots, y_7) \), respectively; then

\[
\begin{align*}
y_0 &= x_1, \\
y_1 &= x_2 \oplus x_0, \\
y_2 &= x_3 \oplus x_0, \\
y_3 &= x_4 \oplus x_0, \\
y_4 &= x_5, \\
y_5 &= x_6 \oplus P(S(x_7 \oplus k)), \\
y_6 &= x_7, \\
y_7 &= x_0 \oplus S(x_7 \oplus k),
\end{align*}
\]

where \( k \) is the round key, \( S \) is the 8-bit to 8-bit Sbox (see [18, 26]), and \( P \) is a linear transformation that permutes the \( i \)-th bit in a byte into position \( P(i) = (1, 7, 5, 4, 2, 6, 0, 3) \).

As our attack is independent of the key schedule and stream cipher, we omit that in this paper. Readers interested are referred to [18, 26] for more details.

For CSA-BC, current works mainly focus on its security evaluation against impossible differential attacks. In [17, 18] In 2016, Zhang et al. [17] proposed the first structural impossible differential attack. Then it is improved to 22 rounds by taking the DDT of the S-box into consideration [18]. The analytical results of CSA-BC are listed in Table 1.

4.2. Impossible Differentials for CSA-BC. In [17, 18], impossible differentials with the pattern

\[
(0, 0, 0, 0, 0, u, 0) \rightarrow (0, v, v, 0, 0, v, u, v \neq 0)
\]

are considered. In this section, we will still study impossible differentials of this type as it diffuses relatively slower. Since more variables are involved in longer impossible differentials, we combined various search strategies as in Section 3, and better results were obtained.

According to Figure 2, one byte is non-linearly transformed after one round; therefore, only one new variable needs to be introduced to describe the differential propagation. From the encryption direction, the 16-round propagation is shown in Table 4, where \( u \in \mathbb{F}_2^* \) and \( u_i \in \mathbb{F}_2^* \), \( 1 \leq i \leq 9 \). The constraints are

\[
\begin{align*}
\text{DDT}[u, u_1] > 0, \text{DDT}[u_1, u_2] > 0, \\
\text{DDT}[u \oplus u_2, u_3] > 0,
\end{align*}
\]
Because one variable is introduced in the beginning and new variables are brought from the 8-th round, there are 
\((r_1 - 6)\) variables in total for an \(r_1\)-round \((r_1 > 7)\) encryption.

Using a similar strategy, the 10-round propagation from the decryption direction could be obtained, and it is shown in Table 5, where \(v \in \mathbb{F}_{2^4}^*\) and \(v_i \in \mathbb{F}_{2^4}, 1 \leq i \leq 3\). The conditions for Sbox are 
\[ DDT[u, v_1] > 0, DDT[v, v_1] > 0, \]
\[ DDT[u_5, u_6] > 0, \]
\[ DDT[u_6, u_7] > 0, DDT[u_6 \oplus Pu_1, u_8] > 0, \]
\[ DDT[u_7 \oplus Pu_2, u_9] > 0. \]  

Because one variable is introduced in the beginning and new variables are brought from the 8-th round, there are 
\((r_1 - 6)\) variables in total for an \(r_1\)-round \((r_1 > 7)\) encryption.

Using a similar strategy, the 10-round propagation from the decryption direction could be obtained, and it is shown in Table 5, where \(v \in \mathbb{F}_{2^4}^*\) and \(v_i \in \mathbb{F}_{2^4}, 1 \leq i \leq 3\). The conditions for Sbox are 
\[ DDT[u, v_1] > 0, DDT[v, v_1] > 0, \]
\[ DDT[u_5, u_6] > 0, \]
\[ DDT[u_6, u_7] > 0, DDT[u_6 \oplus Pu_1, u_8] > 0, \]
\[ DDT[u_7 \oplus Pu_2, u_9] > 0. \]

The number of variables needed to describe the differential propagation for an \(r_2\)-round decryption is also \((r_2 - 6)\) when \(r_2 > 7\).

To search for \(r\)-round \((r = r_1 + r_2\) and \(r \geq 20)\) impossible differentials, \((r - 12)\) 8-bit variables will be introduced. Given the differential state from both sides, 8 equations could be built by assuming they match in all positions, and 8 controlled variables will be deduced correspondingly. Therefore, only \((r - 20)\) bytes need to be guessed instead of \((r - 12)\). At the same time, those variables are input and output differences of the Sbox; hence, they are also required to satisfy the DDT of the Sbox. It should be pointed out that the work of both \([17, 18]\) are special cases of our algorithm.

The 23-round impossible differential is built by concatenating 15 rounds from the encryption direction and 8 rounds from the decryption direction. If they match in every byte, then the equation system for the middle state is built as

\[
\text{(1)} \text{Initiate impossible differential set } T \text{ and possible differential set } T' \text{ to empty} \\
\text{(2)} \text{Derive differential propagation by introducing new variables} \\
\text{(3)} \text{Build equations for the intermediate state} \\
\text{(4)} \text{Represent controlled variables by free variables from the equations} \\
\text{(5)} \text{for all the free variables do} \\
\text{(6)} \text{Compute the controlled variables} \\
\text{(7)} \text{if all conditions of Sbox are satisfied then} \\
\text{(8)} \text{Add this differential to } T' \\
\text{(9)} \text{end if} \\
\text{(10)} \text{end for} \\
\text{(11)} \text{Return } T = (F_8^* \times F_8^* \cap T') \\
\]

**Algorithm 2:** Find impossible differentials with the guess-and-determine technique.

\[
\text{(1)} \text{Initiate impossible differential set } T \text{ to empty} \\
\text{(2)} \text{Derive differential propagation by introducing new variables} \\
\text{(3)} \text{Represent controlled variables by free variables from the equations} \\
\text{(4)} \text{for all the free variables in the first and last round do} \\
\text{(5)} \text{for all remaining free variables do} \\
\text{(6)} \text{if the checking condition of Sbox for free variables is not satisfied then} \\
\text{(7)} \text{Turn to Line 5 } \triangleright \text{ checking process early-aborted} \\
\text{(8)} \text{end if} \\
\text{(9)} \text{if all controlled variables are computed then} \\
\text{(10)} \text{Turn to Line 4 } \triangleright \text{ possible characteristic found, differential early-aborted} \\
\text{(11)} \text{else} \\
\text{(12)} \text{Compute a controlled variable} \\
\text{(13)} \text{end if} \\
\text{(14)} \text{if check condition of Sbox for controlled variables is not satisfied then} \\
\text{(15)} \text{Turn to Line 5 } \triangleright \text{ checking process early-aborted} \\
\text{(16)} \text{else} \\
\text{(17)} \text{Turn to Line 9 } \triangleright \text{ continue checking Sbox conditions} \\
\text{(18)} \text{end if} \\
\text{(19)} \text{end for} \\
\text{(20)} \text{Add this differential into } T \triangleright \text{ Execute this line unless Line10 is executed} \\
\text{(21)} \text{end for} \\
\text{(22)} \text{Return } T \\
\]

**Algorithm 3:** Improved search algorithm with early-abort technique.
| Rounds | 22 (s) | 23 (s) | 24 (s) | 25 | 26 |
|--------|--------|--------|--------|----|----|
| Time taken in Algorithm 2 | <1 | <1 | 27 | 1.698 h | >1d |
| Time taken in Algorithm 3 | <1 | <1 | 3 | 5 s | 3 s |

\[
\begin{aligned}
    \begin{cases}
    u \oplus u_1 \oplus u_2 \oplus u_3 = v_1, \\
    u \oplus u_2 \oplus u_3 \oplus u_4 = 0, \\
    u \oplus u_3 \oplus u_4 \oplus u_5 = v_1, \\
    u \oplus u_4 \oplus u_5 \oplus u_6 = v_1, \\
    u_5 \oplus u_7 = v_1, \\
    u_6 \oplus u_8 = 0, \\
    u \oplus u_7 \oplus u_8 = P v_1, \\
    u \oplus P u_7 \oplus u_8 = v.
    \end{cases}
\end{aligned}
\tag{14}
\]

Since the controlled variables could be expressed as linear functions of free variables, the equation system is transformed by the Gaussian elimination method to

\[
\begin{aligned}
    u_2 &= P^{(236)} \oplus P^{(2467)} v, \\
    v_1 &= P^{-1}[P^{(135)} \oplus P u_1 \oplus P^{(01356)} v], \\
    u_3 &= P^{-1}[P^{(24)} \oplus P u_1 \oplus P^{(02457)} v], \\
    u_4 &= P^{-1}[P^{(02467)} \oplus P u_1 \oplus P^{(023456)} v], \\
    u_5 &= P^{-1}[P^{(127)} \oplus P u_1 \oplus P^{(1267)} v], \\
    u_6 &= P^{(0145)} \oplus P^{(046)} v, \\
    u_7 &= P^{(0135)} \oplus P^{(01356)} v, \\
    u_8 &= P^{(0347)} \oplus P^{(0357)} v, \\
\end{aligned}
\tag{15}
\]

where \(P^{-1}\) is the inverse of \(P\) and \(P^{(N)}\) is the sum of power of \(P\) for short; that is,

\[
P^{(N)} = \oplus_{i \in N} P^i.
\tag{16}
\]

For example, \(P^{(236)} u = \oplus P^{2} \oplus P^{3} \oplus P^{6} u\).

It is deduced that there are 3 free variables \(u, v, u_1\). Given a pair of candidates \((u, v)\), all \(u_1\) should be enumerated to check whether this correspondent characteristic is possible or not. If there exists a set of \((u, v, u_1)\) such that all conditions are satisfied, then \((u, v)\) is a possible differential, and we do not need to guess the remaining candidate \(u_1\). The results from Algorithm 3 show that there are 61814 23-round impossible differentials with input difference \((0, 0, 0, 0, 0, 0, 0, 0)\) and output difference \((0, 0, 0, 0, 0, 0, 0, 0)\).

The 24-round impossible differential is constructed by concatenating 15 rounds forward and 9 rounds backward. The equation system is built and simplified as

\[
\begin{aligned}
    u_2 &= P^{(236)} \oplus P^{(2467)} v, \\
    v_3 &= P^{-1}[P^{(135)} \oplus P u_1 \oplus P^{(01356)} v], \\
    u_3 &= P^{-1}[P^{(24)} \oplus P u_1 \oplus P^{(02457)} v], \\
    u_4 &= P^{-1}[P^{(02467)} \oplus P u_1 \oplus P^{(023456)} v], \\
    u_5 &= P^{-1}[P^{(127)} \oplus P u_1 \oplus P^{(1267)} v], \\
    u_6 &= P^{(0145)} \oplus P^{(046)} v, \\
    u_7 &= P^{(0135)} \oplus P^{(01356)} v, \\
    u_8 &= P^{(0347)} \oplus P^{(0357)} v, \\
\end{aligned}
\tag{17}
\]

Here, the free variables are \(u, v, u_1, v_1, v_2\), and 738 impossible differentials are acquired totally.

The 25-round impossible differential combines 15 rounds from the top and 10 rounds from the bottom. The equation system after simplification is

\[
\begin{aligned}
    u_2 &= P^{(236)} \oplus P^{(2467)} v, \\
    v_3 &= P^{-1}[P^{(135)} \oplus P u_1 \oplus P^{(01356)} v], \\
    u_3 &= P^{-1}[P^{(24)} \oplus P u_1 \oplus P^{(02457)} v], \\
    u_4 &= P^{-1}[P^{(02467)} \oplus P u_1 \oplus P^{(023456)} v], \\
    u_5 &= P^{-1}[P^{(127)} \oplus P u_1 \oplus P^{(1267)} v], \\
    u_6 &= P^{(0145)} \oplus P^{(046)} v, \\
    u_7 &= P^{(0135)} \oplus P^{(01356)} v, \\
    u_8 &= P^{(0347)} \oplus P^{(0357)} v, \\
\end{aligned}
\tag{18}
\]

The free variables are chosen to be \(u, v, u_1, v_1, v_2\). By running through all candidate pairs, only 1 impossible differential is found, which is \((0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)\)\rightarrow\((0, e_8, e_8, e_8, 0, 0, 0, 0, e, 0)\), where the numbers are presented in hexadecimal, and the prefix 0x is omitted.

Note that this method is also applied to 22-round and 26-round. For CSA-BC reduced to 22 rounds, the same results are obtained as in [18], which verified the correctness of our method. For 26-round CSA-BC consisting of 16-round encryption and 10-round decryption, the results confirmed the nonexistence of impossible difference of this considered type.

### 4.3. Effect of Sbox on Impossible Differentials

Since the information on Sbox is involved in our search algorithm for impossible differentials, to verify the dependency of impossible differentials and Sboxes, the Sbox of CSA-BC is replaced by the AES Sbox. AES Sbox [27] is constructed by the inversion function, and it possesses the optimal differential property among 8-bit permutations found so far. The experiment is carried out to the modified cipher, which adopts an identical structure but uses AES Sbox in round function \(F\) instead. Then, the same search procedure is repeated. The comparative results are listed in Table 6.

When the original Sbox of CSA-BC is replaced by the AES Sbox, the number of 23/24/25-round impossible
differentials decreased significantly, which means that our search algorithm is indeed closely related to the detail of the Sbox. An example is that $(0, 0, 0, 0, 0, 0, e4, 0) \rightarrow (0, e8, e8, e8, 0, 0, 0, e8)$ is a 24-round impossible differential for CSA. However, it is possible when the Sbox is replaced by the AES Sbox. Since there are few results on Sbox-related impossible differential, our method may be a possible tool to search for Sbox-related impossible differentials of similar block ciphers. In addition, as far as the number of impossible differentials is concerned, replacing the Sbox of CSA-BC with AES Sbox might improve its resistance against impossible differential attack for CSA-BC.

4.4. Key Recovery Attacks against 25-Round CSA-BC and CSA.

Hereinbefore, impossible differential distinguishers have been constructed for CSA-BC reduced to 23/24/25 rounds. As there is only one impossible differential for 25 rounds, to achieve better attack performance, we will mount the key recovery attack against 25-round CSA-BC and CSA using the 24-round impossible differentials found above.

As is presented in Figure 3, 1 round is appended at the end of the 24-round impossible differentials. The detailed attack procedure is as follows:

(1) Define a plaintext structure whose 7-th byte takes all possible values and others constant

$$\Lambda = \{(c_0, c_1, c_2, c_3, c_4, c_5, c_6) | x \in F_{2^8}\},$$

where $c_i (0 \leq i \leq 6)$ are constants over $F_{2^8}$. For any two plaintexts in $\Lambda$, their difference will be of the form $(0, 0, 0, 0, 0, 0, *, 0)$.

(2) Choose $2^N$ structures, and then $2^{N+15}$ pairs will be generated with the desired input difference. Choose pairs whose difference of the ciphertext is

$$(v, v, v, 0, 0, P(w), v, w), v, w \in F_{2^8}.$$  \hspace{1cm} (20)

The probability that the difference of ciphertext pairs is of this expected form is $2^{-48}$. As there are 738 24-round impossible differentials, that is, 738 $(u, v)$ candidates, the proportion of pairs in accordance with the input and output difference of the attack scenario is $738/2^{16} \approx 2^{-6.47}$. Therefore, the probability that a plaintext pair from Step (1) results in the desired difference in ciphertext is $2^{-48} \times 2^{-6.47} = 2^{-54.47}$, and the number of pairs satisfying the attack scenario is $2^{N+15} \times 2^{-54.47} \approx 2^{N-39.47}$. \hspace{1cm} (21)

(3) Guess the round keys in the last round and decrypt for 1 round. If the difference after decryption is $(0, v, v, v, 0, 0, v)$, discard this key candidate and try another. Repeat this process until only the right key is left. The remaining key bits could be recovered by brute force attack.

The probability that a wrong key leads to the required difference in the key recovery process is $(1 - 2^{-8})$; then there would be $(2^8 - 1) \cdot (1 - 2^{-8})^{25-1} = 2^{-48}$ wrong keys left. When we take $N = 50.5$, the expected wrong key left after filtering is approximately 0.07. In this case, all wrong keys are thought to be removed. Hence, the data complexity is $2^{50.5} \times 2^8 = 2^{58.5}$ chosen plaintexts. According to the quick-sort method, the time complexity of the data filtration phase is $2^{50.5} \times 2^8 \times \log_2 (2^8) = 2^{61.5}$ comparison. In the worst case, $2^{N-39.47} \times 2^8 = 2^{20.03}$ 1-round decryption will be needed, which is neglectable compared with the date filtration phase. As a counter is used in data filtration and should be kept for the key candidate and the pairs after filtration, the memory complexity is $2^8 + 2^{11.65} + 2^8 \approx 2^{11.35}$ memory units.

As is pointed out in [17], a structural flaw could be exploited to extend the attack of block cipher CSA-BC to the whole algorithm CSA. Here, a similar method could be adopted where the first block is active and others are inactive. Choose the plaintext of the first block as in Step (1) and mount the same procedures of its ciphertext as in Steps (2) and (3); the keys could be recovered with data complexity $2^{58.5}$ chosen plaintexts, time complexity $2^{61.5}$ comparison, and memory complexity $2^{11.35}$ memory units. The results are listed in Table 1.

5. Constructing Impossible Differentials for FOX64

In this section, we will first introduce the FOX64 block cipher, then our search method will be applied, and the improved results will be discussed.

5.1. FOX64 Block Cipher. The block cipher FOX, also known as IDEA-NXT, was proposed by Junod and Vaudenay in 2004 [28]. It was designed to have a high-security level, large implementation flexibility, and high performance on various platforms. FOX employs the Lai-Massey structure and an SPS-type round function, and it proved to be immune to differential and linear attacks. The 64-bit block length and 128-bit key length version of FOX is called FOX64, and it has 16 rounds in total.

FOX64 employs the Lai-Massey structure, as is shown in Figure 4. Denote the input and output of FOX64 as $(x_1, x_2), (y_1, y_2)$, respectively; then

$$\begin{align*}
y_1 &= ol(x_1 \oplus F(x_1 \oplus x_2, rk)), \\
y_2 &= x_1 \oplus F(x_1 \oplus x_2, rk),
\end{align*}$$

Figure 3: Key recovery attack of 25-round CSA-BC.
where \( r_k \) is the round key. The orthomorphism or uses a 1-round Feistel transformation. Denote the 32-bit input as \((l, r) \in \mathbb{F}_{28}; \) then

\[
or (l, r) = (r, l \oplus r).
\]  

(22)

The round function \( F \) employs an SPS-type design. It is composed of three parts: substitution part (\( \text{sigma4} \)), diffusion part (\( \text{mu4} \)), and round key addition part. The round function \( F \) takes a 32-bit input \( x \) and a 64-bit round key \( r_k = r_k_1 \oplus r_k_2 \); then

\[
F(x, r_k) = \text{sigma4}(\text{mu4}(\text{sigma4}(x \oplus r_k_1)) \oplus r_k_2) \oplus r_k_1.
\]  

(23)

The substitution layer consists of 4 identical 8-bit Sboxes. The detailed permutation of the Sbox and its construction is discussed in [28]. The diffusion layer is a matrix multiplication over a finite field. The 32-bit input is regarded as a 4-dimensional vector over \( \mathbb{F}_{28} \), and the \( 4 \times 4 \) matrix is

\[
\text{mu4} = \begin{bmatrix}
1, 1, 1, \alpha \\
1, \alpha, 1, 1 \\
\alpha, \alpha, 1, 1 \\
\alpha, 1, \alpha, 1
\end{bmatrix},
\]  

(24)

where \( \mathbb{F}_{28} = \mathbb{F}_2(\alpha) \), \( \alpha \) is a root of polynomial \( x^8 + x^4 + x^3 + x^2 + x + 1 \), and \( z = \alpha^3 + 1 \).

The security of FOX64 and its high-level structure are extensively studied [29–32]. For the impossible differential attack, Wu et al. [33] presented an impossible differential for 4-round FOX64 of the following form \((0, 0, 0, 0, 0, 0) \rightarrow (v_1, v_2, v_3, v_4, v_1, v_2)\). This distinguisher is further improved and generalized in [34]. In 2016, Li et al. used multiple impossible differentials and various techniques to mount an 8-round key recovery attack using similar 4-round impossible differentials [35]. The analytical results of FOX are listed in Table 2.

### 5.2. Impossible Differentials for FOX64

The impossible differentials are constructed using the method in Section 3. Using Algorithm 3, the relationship between FOX64 and its round function \( F \) is first observed as the following theorem by exploiting the middle state conditions.

**Theorem 1.** Verifying a differential of the following form for 4-round FOX64

\[
(u_1, u_2, u_3, u_4, u_1, u_2, u_3, u_4) \rightarrow 4-\text{round-FOX64} (v_1, v_2, v_3, v_4, v_1, v_2, v_3, v_4)
\]  

is equivalent to verifying a differential of its round function where \( u_i, v_i \in \mathbb{F}_{28} \). Furthermore, an impossible differential for its round function will always imply an impossible differential for 4-round FOX64.

**Proof.** First, the differential propagation of FOX64 is presented in Figure 5.

For any input difference \((u_1, u_2, u_3, u_4, u_1, u_2, u_3, u_4) \in \mathbb{F}_{28}^8\), the output difference after 2-round encryption is

\[
(u_1 \oplus u_2 \oplus u_1, u_2 \oplus u_4 \oplus u_4) \oplus (u_1 \oplus u_4 \oplus u_2, u_2 \oplus u_4 \oplus u_2) \oplus (u_2 \oplus u_4 \oplus u_2, u_2 \oplus u_4 \oplus u_4).
\]
FOX64, those two independent parts. According to the structure of variables, intermediate variables produces 8-byte controlled FOX64 is possible or not, this relationship among in-

equivalent to verifying one differential of its round function.

The 4-round impossible differentials are constructed by combining 2-round encryption and 2-round decryption. As possible differentials will match in all positions in the middle state, an equation system could be built reflecting this internal relationship

\[
\begin{align*}
&u_1 \oplus u_3 \oplus u_{11} = v_1 \oplus v_3 \oplus v_9, \\
&u_2 \oplus u_4 \oplus u_{12} = v_2 \oplus v_4 \oplus v_{10}, \\
&u_1 \oplus u_3 \oplus u_{11} = v_1 \oplus v_{11}, \\
&u_2 \oplus u_4 \oplus u_{12} = v_2 \oplus v_{12}, \\
&u_1 \oplus u_{10} = v_1 \oplus v_{10}, \\
&u_2 \oplus u_{11} = v_2 \oplus v_{11}, \\
&u_4 \oplus u_{12} = v_4 \oplus v_{12},
\end{align*}
\]

(26)

This equation system could be simplified by the Gaussian elimination method as

\[
\begin{align*}
&u_9 = u_1 \oplus u_3 \oplus v_1 \oplus v_3, \\
&u_{10} = u_2 \oplus u_4 \oplus v_2 \oplus v_4, \\
&u_{11} = u_1 \oplus v_1, \\
&u_{12} = u_2 \oplus v_2, \\
&v_9 = u_3 \oplus v_3, \\
&v_{10} = u_4 \oplus v_4, \\
&v_{11} = u_3 \oplus u_4 \oplus v_3 \oplus v_5, \\
&v_{12} = u_4 \oplus u_3 \oplus v_2 \oplus v_4.
\end{align*}
\]

(27)

When verifying whether a 4-round differential of FOX64 is possible or not, this relationship among intermediate variables produces 8-byte controlled variables.

The determining process could be further separated into two independent parts. According to the structure of FOX64, those \( u_i \)'s are only involved in the second round function, and \( v_i \)'s are only involved in the third. Therefore, the determining process could be divided into checking the validity of the following two differentials for the round function \( F \)

\[
\{(u_1 \oplus u_3, u_2 \oplus u_4, u_1, u_2) \}
\]

\( \xrightarrow{F} (u_9, u_{10}, u_{11}, u_{12}) , (v_3, v_4, v_1 \oplus v_3, v_2 \oplus v_4) \)

\( \xrightarrow{F} (v_9, v_{10}, v_{11}, v_{12}) \).

(28)

That is, verifying a differential for 4-round FOX64 is equivalent to verifying one differential of its round function.

If the differential is impossible for \( F \), it implies one impossible differential for 4-round FOX64.

**Remark 3.** In Theorem 1 and its proof, the relationship between 4-round FOX and its round functions is derived by exploiting the middle state conditions only. It indicates when an impossible differential for \( F \) is detected, it could be extended to FOX64 by equations (27) and (28). Then the search for impossible differential of 4-round FOX is transformed to checking the possibility of differentials of round function \( F \), which corresponds to the Sbox conditions in Algorithm 3, and the early-abort technique will also work.

According to equation (27), the controlled variables are \( u_i, v_i (9 \leq i \leq 12) \), the free variables in the difference pattern are \( u_i, v_i (1 \leq i \leq 4) \), and the remaining free variables are \( u_6, v_7 (5 \leq i \leq 8) \). To check the possibility of \((u_1 \oplus u_{13}, u_2 \oplus u_{14}, u_1, u_2) \xrightarrow{F} (u_9, u_{10}, u_{11}, u_{12}) \) as in equation (28), all \( u_i, v_i (5 \leq i \leq 8) \) are required to be enumerated. Assuming the difference after \( muf \) is \((x_1, x_2, x_3, x_4) = muf (u_5, u_6, u_7, u_8) \), then the Sbox conditions are

\[
\begin{align*}
DDT[u_1 \oplus u_3, u_5] &> 0, DDT[u_2 \oplus u_4, u_6] > 0, \\
DDT[u_1, u_5] &> 0, DDT[u_2, u_6] > 0, \\
DDT[x_1, u_5] &> 0, DDT[x_1, u_{10}] > 0, \\
DDT[x_3, u_{11}] &> 0, DDT[x_4, u_{12}] > 0.
\end{align*}
\]

(29)

When all those Sbox conditions are satisfied, then this characteristic is possible, indicating the 4-round differential is also possible. And this differential is impossible only when at least one Sbox condition is not satisfied for each characteristic, that is, each set of \( u_i, v_i (5 \leq i \leq 8) \).

This theorem indicates that the search of impossible differentials for FOX64 is equivalent to that of its round function. The complexity is reduced to \( O(2^{23}) \), which is practical for current personal computers. The experimental results indicate that verifying one difference for FOX64 requires approximately \( 10^{-6} \) seconds on average. Moreover, previous works could also be obtained by our method.

Using the above method, we are capable of finding new impossible differentials for FOX64 with a more freedom degree. The results show that the number of impossible differentials of the following form for round function \( F \)

\[(01, 00, 00, 00) \rightarrow (v_1, v_2, v_3, v_4), \]

(30)

where \( v_i \in F_2 \), is 1197321873, accounting for 28.31% of the whole space approximately. And the time taken is 2133 seconds. When the input difference of \( F \) is \((01, 02, 00, 00) \), this proportion is reduced to \( 2^{-18.85} \). 8941 impossible differentials in total.

After acquiring the impossible differential for round function \( F \), the impossible differential for 4-round FOX64 could be constructed accordingly. For example, \((01, 02, 00, 00) \rightarrow (ff, fa, 11, cc) \) is an impossible differential for \( F \); then
is a 4-round impossible differential for FOX64.

\begin{table}[h]
\centering
\caption{Differential propagation from encryption direction.}
\begin{tabular}{c|c}
\hline
Differential state & \\
\hline
0 & (0,0,0,0,0,0,0,0) \\
1 & (0,0,0,0,0,0,0,0) \\
2 & (0,0,0,0,0,0,0,0) \\
3 & (0,0,0,0,0,0,0,0) \\
4 & (0,0,0,0,0,0,0,0) \\
5 & (0,0,0,0,0,0,0,0) \\
6 & (0,0,0,0,0,0,0,0) \\
7 & (u,v,u,u,u,u,u,u) \\
8 & (u,u,u,u,u,u,u,u) \\
9 & (u,u,u,u,u,u,u,u) \\
10 & (0,0,0,0,0,0,0,0) \\
11 & (0,0,0,0,0,0,0,0) \\
12 & (0,0,0,0,0,0,0,0) \\
13 & (0,0,0,0,0,0,0,0) \\
14 & (0,0,0,0,0,0,0,0) \\
15 & (0,0,0,0,0,0,0,0) \\
16 & (0,0,0,0,0,0,0,0) \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\caption{Differential propagation from decryption direction.}
\begin{tabular}{c|c}
\hline
Differential state & \\
\hline
0 & (0,0,0,0,0,0,0,0) \\
1 & (0,0,0,0,0,0,0,0) \\
2 & (0,0,0,0,0,0,0,0) \\
3 & (0,0,0,0,0,0,0,0) \\
4 & (0,0,0,0,0,0,0,0) \\
5 & (0,0,0,0,0,0,0,0) \\
6 & (0,0,0,0,0,0,0,0) \\
7 & (0,0,0,0,0,0,0,0) \\
8 & (0,0,0,0,0,0,0,0) \\
9 & (0,0,0,0,0,0,0,0) \\
10 & (0,0,0,0,0,0,0,0) \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\caption{Number of impossible differentials for CSA-BC with different Sbox.}
\begin{tabular}{|c|c|c|c|}
\hline
Distinguisher length & 23 & 24 & 25 \\
\hline
CSA-BC & 61814 & 738 & 1 \\
CSA-BC with AES Sbox & 51337 & 478 & 0 \\
\hline
\end{tabular}
\end{table}

6. Conclusion

In this paper, we proposed a new Sbox-related impossible differential search method combining the guess-and-determine and early-abort techniques. Our method is able to exploit more detailed information on Sboxes, and the impossible differentials found are closely related to the choice of Sboxes. Besides, our method is independent of a third-party solver and applicable to ciphers with Sboxes of large size (≥8 bits).

Then this method is applied to CSA-BC and FOX64. With our method, 23/24/25-round impossible differentials for CSA-BC were found in less than 5 seconds. Those results are the longest impossible differentials for CSA-BC so far. And key recovery attacks were improved by 4 rounds using the impossible differentials found by our method. Those key recovery attacks are the longest impossible differential attack against CSA-BC and CSA by now. For FOX64, we first proved that an impossible differential for its round function will always imply one impossible differential for 4-round
FOX64. Then new types of impossible differentials were able to be found with our method.

Our results may help better understand the interaction between the structure and primitives of block ciphers. And the search method in this paper may provide a possible direction to take the large size of Sboxes into consideration and is applicable to other block ciphers with a similar structure.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (nos. 62172427, 62002370, and 61702537), the Scientific Research Plan of National University of Defense Technology (no. ZK21-36), and the State Key Laboratory of Information Security (no. 2020-MS-02).

References

[1] L. Knudsen, DEAL - A 128-bit Block Cipher, Nist Aes Proposal edition, 1998.
[2] E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials,” in Proceedings of the Advances in Cryptology - EUROCRYPT’99, Prague, Czech Republic, pp. 12–23, Springer, Berlin, Heidelberg, May 2–6(1999).
[3] W. Zhang, W. Wu, and D. Feng, "New results on impossible differential cryptanalysis of reduced AES,” in Proceedings of the Information Security and Cryptology - ICISC 2007, Seoul, Korea, pp. 239–250, Springer, Berlin, Heidelberg, November 29–30(2007).
[4] S. A. Azimi, S. Ahmadi, Z. Ahmadian, J. Mohajeri, and M. R. Aref, "Improved impossible differential and biclique cryptanalysis of HIGHT,” International Journal of Communication Systems, vol. 31, no. 1, p. e3382, 2018.
[5] J. Kim, S. Hong, J. Sung, C. Lee, and S. Lee, "Impossible differential cryptanalysis for block cipher structures,” in Proceedings of the Progress in Cryptology - INDOCRYPT 2003, New Delhi, India, pp. 82–96, Springer, Berlin, Heidelberg, December 8–10(2003).
[6] Y. Luo, X. Lai, Z. Wu, and G. Gong, "A unified method for finding impossible differentials of block cipher structures,” Information Sciences, vol. 263, pp. 211–220, 2014.
[7] S. Wu and M. Wang, "Automatic search of truncated impossible differentials for word-oriented block ciphers,” in Proceedings of the Progress in Cryptology - INDOCRYPT 2012, Kolkata, India, pp. 283–302, Springer, Berlin, Heidelberg, December 9–12(2012).
[8] C. Blondeau and M. Minier, “Analysis of impossible, integral and zero-correlation attacks on type-ii generalized feistel networks using the matrix method,” in Proceedings of the Fast Software Encryption - FSE 2015, Istanbul, Turkey, pp. 92–113, Springer, Berlin, Heidelberg, March 8–11 (2015).
[9] T. P. Berger and M. Minier, “Some results using the matrix methods on impossible, integral and zero-correlation distinguishers for feistel-like ciphers,” in Proceedings of the Progress in Cryptology - INDOCRYPT 2015, Bangalore, India, pp. 180–197, Springer, Cham, December 6–9 (2015).
[10] X. Shen, R. Li, B. Sun, L. Cheng, C. Li, and M. Liao, “Dual relationship between impossible differentials and zero correlation linear hulls of Simonlike ciphers,” in Proceedings of the Information Security Practice and Experience - ISPEC 2017, Melbourne, VIC, Australia, pp. 237–255, Springer, Cham, December 13–15(2017).
[11] N. Mouha, Q. Wang, D. Gu, and B. Preneel, “Differential and linear cryptanalysis using mixed integer linear programming,” in Proceedings of the Information Security and Cryptology - Inscrypt 2011, Beijing, China, pp. 57–76, Springer, Berlin, Heidelberg, November 30–December 3(2011).
[12] Z. Xiang, W. Zhang, Z. Bao, and D. Lin, “Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers,” in Proceedings of the Advances in Cryptology - ASIACRYPT 2016, Hanoi, Vietnam, pp. 648–678, Springer, Berlin, Heidelberg, December 4–8(2016).
[13] D. Shi, S. Sun, P. Derbez, Y. Todo, B. Sun, and L. Hu, "Programming the demirci-selçuk meet-in-the-middle attack with constraints,” in Proceedings of the Advances in Cryptology - ASIACRYPT 2018, Brisbane, QLD, Australia, pp. 3–34, Springer, Cham, December 2–6(2018).
[14] Y. Liu, S. Sun, and C. Li, “Rotational cryptanalysis from a differential-linear perspective - practical distinguishers for round-reduced friet, xoodoo, and azette,” in Proceedings of the Advances in Cryptology - EUROCRYPT 2021, Zagreb, Croatia, pp. 741–770, Springer, Cham, October 17–21(2021).
[15] Y. Sasaki and Y. Todo, “New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers,” in Proceedings of the Advances in Cryptology - EUROCRYPT 2017, Paris, France, pp. 185–215, Springer, Cham, April 30–May 4(2017).
[16] X. Hu, Y. Li, L. Jiao, S. Tian, and M. Wang, "Mind the propagation of states - new automatic search tool for impossible differentials and impossible polytopic transitions,” in Proceedings of the Advances in Cryptology - ASIACRYPT 2020, Daejeon, South Korea, pp. 415–445, Springer, Cham, December 7–11(2020).
[17] K. Zhang, J. Guan, and B. Hu, "Impossible differential cryptanalysis on DVB-CSA,” KSII Trans. Internet Inf. Syst., vol. 10, pp. 1944–1956, 2016.
[18] X. Shen, B. Sun, G. Liu, and C. Li, “Impossible differential cryptanalysis of the digital video broadcasting-common scrambling algorithm,” Journal of Electronics and Information Technology, vol. 41, pp. 46–52, 2019.
[19] L. Jiao, Y. Li, and Y. Hao, "A guess-and-determine attack on SNOW-V stream cipher,” The Computer Journal, vol. 63, no. 12, pp. 1789–1812, 2020.
[20] X. Feng, J. Liu, Z. Zhou, C. Wu, and D. Feng, "A byte-based guess and determine attack on SOSEMANUK,” in Proceedings of the Advances in Cryptology - ASIACRYPT 2010, Singapore, pp. 146–157, Springer, Berlin, Heidelberg, December 5–9(2010).
[21] Y. Wei, E. Pasalic, and Y. Hu, “Guess and determine attacks on filter generators revisited,” IEEE Transactions on Information Theory, vol. 58, no. 4, pp. 2530–2539, 2012.
[22] E. Biham and A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer, New York, NY, 1993.
[23] C. Boura, V. Lallemand, M. Naya-Plasencia, and V. Suder, “Making the impossible possible,” Journal of Cryptology, vol. 31, no. 1, pp. 101–133, 2018.

[24] M. Fouquet, P. Gaudry, and R. Harley, “Finding secure curves with the Satoh-FGH algorithm and an early-abort strategy,” in Proceedings of the Advances in Cryptology - EUROCRYPT 2001, Innsbruck, Austria, pp. 14–29, Springer, Berlin, Heidelberg, May 6–10(2001).

[25] C. Cid, S. Murphy, and B. Robshaw, “Small scale variants of the AES,” Fast Software Encryption, pp. 145–162, 2005.

[26] R. Weinmann and K. Wirt, “Analysis of the DVB common scrambling algorithm,” in Proceedings of the Communications and Multimedia Security, Windermere, United Kingdom, pp. 195–207, Springer, Boston, MA, Sept. 15–18 2005.

[27] J. Daemen and V. Rijmen, The Design of Rijndael - the Advanced Encryption Standard (AES), Springer, Berlin, Heidelberg, Second Edition, 2020.

[28] P. Junod and S. Vaudenay, “FOX: a new family of block ciphers,” in Proceedings of the Selected Areas in Cryptography, SAC 2004, Waterloo, pp. 114–129, Springer, Berlin, Heidelberg, Canada, August 9–10 2004.

[29] T. Isobe and K. Shibutani, “Improved all-subkeys recovery attacks on fox, KATAN and SHACAL-2 block ciphers,” in Proceedings of the Fast Software Encryption - FSE 2014, London, UK, pp. 104–126, Springer, Berlin, Heidelberg, March 3–5 2014.

[30] K. Qiao, L. Hu, S. Sun, X. Ma, and H. Kan, “Improved MILP modeling for automatic security evaluation and application to FOX,” IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E98.A, no. 1, pp. 72–80, 2015.

[31] R. Li, J. You, B. Sun, and C. Li, “Fault analysis study of the block cipher FOX64,” Multimedia Tools and Applications, vol. 63, no. 3, pp. 691–708, 2013.

[32] Y. Luo, X. Lai, and Y. Zhou, “Generic attacks on the lai-massey scheme,” Designs, Codes and Cryptography, vol. 83, no. 2, pp. 407–423, 2017.

[33] Z. Wu, Y. Luo, X. Lai, and B. Zhu, “Improved cryptanalysis of the FOX block cipher,” in Proceedings of the Trusted Systems, INTRUST 2009, Beijing, China, pp. 236–249, Springer, Berlin, Heidelberg, December 17–19 2009.

[34] R. Guo and C. Jin, “Impossible differential cryptanalysis on lai-massey scheme,” ETRI Journal, vol. 36, no. 6, pp. 1032–1040, 2014.

[35] X. Li, C. H. Jin, and F. W. Fu, “Improved results of impossible differential cryptanalysis on reduced FOX,” The Computer Journal, vol. 59, no. 4, pp. 541–548, 2016.