Research Article
Cryptanalysis of the Lightweight Block Cipher BORON

Huicong Liang\textsuperscript{1,2} and Meiqin Wang\textsuperscript{1,2}

\textsuperscript{1}School of Cyber Science and Technology, Shandong University, Qingdao, China
\textsuperscript{2}Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao, China

Correspondence should be addressed to Meiqin Wang; mqwang@sdu.edu.cn

Received 15 March 2019; Revised 1 November 2019; Accepted 25 November 2019; Published 18 December 2019

Copyright © 2019 Huicong Liang and Meiqin Wang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper provides security evaluations of a lightweight block cipher called BORON proposed by Bansod et al. There is no third-party cryptanalysis towards BORON. Designers only provided coarse and simple security analysis. To fill this gap, security bounds of BORON against differential and linear cryptanalysis are presented in this paper. By automatic models based on the SMT solver STP, we search for differential and linear trails with the minimal number of active S-boxes and trails with optimal probability and bias. Then, we present key-recovery attacks towards round-reduced BORON. This paper is the first third-party cryptanalysis towards BORON.

1. Introduction

Lightweight cryptography is one of the most actively discussed topics in the current cryptographic community. For the last decade, a great number of lightweight block ciphers have been proposed, such as PRESENT \cite{1}, CLEFIA \cite{2}, PRINCE \cite{3}, SIMON \cite{4}, SPECK \cite{4}, SKINNY \cite{5}, GIFT \cite{6}, QARMA \cite{7}, and so on, which draw plenty of researchers’ attention.

Meanwhile, there are some unknown lightweight block ciphers. BORON \cite{8} is one of them. BORON, a substitution-permutation network cipher, supports 64-bit messages and 80/128-bit keys, which has a total of 25 rounds. The substitution layer is made up of 16 same 4-bit S-Boxes, and the permutation layer includes nibble shift, bit rotation, and block XOR operation.

Except for simple security analysis illustrated by designers, there is no third-party security evaluation in the open literature. To fill this gap, differential \cite{9} and linear cryptanalysis \cite{10} towards BORON are presented in this paper. Differential and linear cryptanalysis are one of the most fundamental and powerful techniques for security evaluations of cryptographic primitives.

Differential cryptanalysis discusses how differences propagate through the objective cryptographic primitives. If input differences and output differences occur in some regular patterns, these could be used to build distinguishers or even to recover secret keys.

Linear cryptanalysis focuses on the linear equation between plaintexts, ciphertexts, and keys. If this linear equation happens with a high probability, the distinguishing attack or key-recovery attack could be presented. Therefore, finding good distinguishers is the first step to evaluate security against differential and linear cryptanalysis.

There are many methods to find differential and linear trails, such as Matsui’s algorithm, SAT/SMT and MILP, which are widely used to analyse the security of cryptographic primitives \cite{11–15}. In this paper, we utilize the SMT solver STP (http://stp.github.io) to search for good trails, which is suitable for dealing with bit vectors.

1.1. Contributions. This paper analyses the security of BORON against differential and linear cryptanalysis. Security bounds are depicted in Table 1.

The minimal number of active S-boxes of differential and linear trails could describe approximate security bounds resistance to differential and linear cryptanalysis. We find that the minimal number of active S-boxes proposed by
designers is incorrect and intact. Then, differential and linear trails with the accurate minimal number of active S-boxes are found. Our results are listed in Table 2.

We also search for (related-key) differential and linear trails with the optimal probability and bias. Details about the optimal probability and bias are illustrated in Table 3. Specific trails with the optimal probability and bias are listed in Tables 4–6. We figure out that there is no effective 9-round differential trail and 10-round linear trail. We also try to find impossible differential trails of BORON, and 7-round BORON impossible differential trails are obtained.

Utilizing the 8-round differential trail with the optimal probability of $2^{-62}$, we depict a key-recovery attack towards 9-round BORON with the 80-bit key schedule, whose time complexity is $2^{56}$, data complexity is $2^{63}$, and memory complexity is $2^{24}$. By the 9-round linear trail with the optimal bias of $2^{-30}$, a key-recovery attack towards 11-round BORON with the 128-bit key schedule is described, whose time complexity is $2^{123}$, data complexity is $2^{65}$, and memory complexity is $2^{42}$.

And, in this paper, we assume that BORON is the Markov cipher.

1.2. Outline. This paper is organized as follows. Section 2 introduces the description of BORON. In Section 3, automatic models for finding trails are illustrated. Key-recovery attacks towards BORON are presented in Section 4. Section 5 concludes this paper.

2. Description of BORON

2.1. Encryption Algorithm. BORON is a 25-round SPN block cipher, whose message length is 64-bit and the key length is 80-bit or 128-bit. The round function of BORON is made up of Substitution Layer and Permutation Layer. The Substitution Layer consists of 16 same 4-bit S-boxes in parallel. The S-box is described in Table 7. The Permutation Layer includes three parts, nibble shift, bit rotation, and block XOR operation. And, the round function is described in Figure 1. For more details, refer [8].

In Figure 1, we regard 4 bits as a nibble and 16 bits as a block. Note that nibble shift is the permutation between nibbles, while bit rotation and block XOR operation are ones in term of blocks.

2.2. Key Schedule. The key schedule of BORON is inspired by the one of PRESENT [1]. There are two versions, 80-bit and 128-bit. In total, 26 subkeys are generated by the key schedule in the whole procedure of encryption. Each subkey has 64 bits.

For the 80-bit version, the master key could be stored in a key register and denoted as $K = k_78 \ldots k_0$. First, extract the least significant 64 bits of the master key as the first subkey, that is, the whitening key $K_0 = k_63k_62 \ldots k_0$. In the $j$th round ($j = 0, 1, \ldots, 24$), after extracting the least significant 64 bits, the key register is updated as follows:

1. $K \leftarrow 13$
2. $k_0 || k_6 || k_0 \leftarrow S[k_3 || k_2 || k_1 || k_0]$
3. $k_{63} || k_{62} || k_{61} \leftarrow S[k_{65} || k_{64} || k_{63} || k_{62} || k_{61} || k_{60} || k_{59}] @ j$

For the 128-bit version, the master key could be stored in a key register and denoted as $K = k_{127} \ldots k_0$. Extract the least significant 64 bits of the master key as the first subkey, that is, the whitening key $K_0 = k_{63}k_{62} \ldots k_0$. In the $j$th round ($j = 0, 1, \ldots, 24$), after extracting the least significant 64 bits, the key register is updated as follows:

1. $K \leftarrow 13$
2. $k_0 || k_6 || k_0 \leftarrow S[k_3 || k_2 || k_1 || k_0]$
3. $k_{63} || k_{62} || k_{61} \leftarrow S[k_{65} || k_{64} || k_{63} || k_{62} || k_{61} || k_{60} || k_{59}] @ j$
4. $k_{63} || k_{62} || k_{61} \leftarrow S[k_{65} || k_{64} || k_{63} || k_{62} || k_{61} || k_{60} || k_{59}] @ j$

3. Automatic Models for Searching Trails

In this paper, we use STP [16], an SMT solver, to search for good differential and linear trails. STP uses CVC and SMTLIB2 languages to encode constraints and then invokes an SAT solver to check for the satisfiability of these constraints. In this paper, the CVC language is used to encode difference and linear mask propagations of BORON.
Table 4: Differential trails with the optimal probability.

<table>
<thead>
<tr>
<th>Round</th>
<th>Input difference of S-box</th>
<th>Output difference of S-box</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-round differential trail with probability 2−12</td>
<td>0x0000000080000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>0</td>
<td>0x0000000080000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0x1000100000000000</td>
<td>0xA000A00000000000</td>
</tr>
<tr>
<td>3</td>
<td>0x1141514040014141</td>
<td>—</td>
</tr>
<tr>
<td>4-round differential trail with probability 2−19</td>
<td>0xA000000090000000</td>
<td>0x0000000010000000</td>
</tr>
<tr>
<td>1</td>
<td>0x0000008000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0x6006000000000000</td>
<td>0x2000800000000000</td>
</tr>
<tr>
<td>3</td>
<td>0x0000004000400400</td>
<td>0x0000009000000700</td>
</tr>
<tr>
<td>4</td>
<td>0x48E048E090009E0</td>
<td>—</td>
</tr>
<tr>
<td>5-round differential trail with probability 2−28</td>
<td>0x8A018A01B0000A00</td>
<td>0x0000000030000000</td>
</tr>
<tr>
<td>0</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>4</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>5</td>
<td>0x8A018A01B0000A00</td>
<td>0x0000000030000000</td>
</tr>
</tbody>
</table>

Table 5: Linear trails with the optimal bias.

<table>
<thead>
<tr>
<th>Round</th>
<th>Input mask of S-box</th>
<th>Output mask of S-box</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-round linear trail with bias 2−7</td>
<td>0x000D000000000000</td>
<td>0x0800000000000100</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000300</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000300</td>
</tr>
<tr>
<td>4-round linear trail with bias 2−11</td>
<td>0x0D00000000000000</td>
<td>0x8000000000000100</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000300</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000300</td>
</tr>
<tr>
<td>3</td>
<td>0x0010010000000000</td>
<td>0x0030003000000070</td>
</tr>
<tr>
<td>5-round linear trail with bias 2−15</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>6-round linear trail with bias 2−19</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>4</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>7-round linear trail with bias 2−23</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>4</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>8-round linear trail with bias 2−27</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>4</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>9-round linear trail with bias 2−30</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>4</td>
<td>0x0000000000000000</td>
<td>0x0000000000000000</td>
</tr>
</tbody>
</table>

Inspired by models in [11–15, 17–20], we present automatic models for finding differential and linear trails of BORON. All intermediate states of BORON are represented by variables which are used to build constraints. Searching for good trails is transformed into checking for the satisfiability of constraints. For different searching problems, there are different constraints. We show constraints for all operations used in BORON as follows.

3.1. Finding Differential Trails. First, we give some notations about variables. We count from the right side and zero.

(i) before_sbox_value1_i_j: input value of the jth S-box in the ith round
(ii) after_sbox_value1_i_j: output value of the jth S-box in the ith round whose input value is before_sbox_value1_i_j
(iii) before_sbox_value2_i_j: input value of the jth S-box in the ith round
after_sbox_value_{i,j}: output value of the jth S-box in the i-th round whose input value is before_sbox_value_{i,j}

<table>
<thead>
<tr>
<th>Round</th>
<th>Subkey difference</th>
<th>Input difference of S-box</th>
<th>Output difference of S-box</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-round related-key differential trail with probability $2^{-2}$</td>
<td>0 (0x0020000000000000)</td>
<td>0(x0000000000000000)</td>
<td>0(x0000000000000000)</td>
</tr>
<tr>
<td>5-round related-key differential trail with probability $2^{-6}$</td>
<td>0 (0x0000000000000000)</td>
<td>0(x0000000000000000)</td>
<td>0(x0000000000000000)</td>
</tr>
<tr>
<td>6-round related-key differential trail with probability $2^{-15}$</td>
<td>0 (0x0000000000000000)</td>
<td>0(x0000000000000000)</td>
<td>0(x0000000000000000)</td>
</tr>
<tr>
<td>7-round related-key differential trail with probability $2^{-27}$</td>
<td>0 (0x0000000000000000)</td>
<td>0(x0000000000000000)</td>
<td>0(x0000000000000000)</td>
</tr>
<tr>
<td>8-round related-key differential trail with probability $2^{-40}$</td>
<td>0 (0x0000000000000000)</td>
<td>0(x0000000000000000)</td>
<td>0(x0000000000000000)</td>
</tr>
<tr>
<td>9-round related-key differential trail with probability $2^{-48}$</td>
<td>0 (0x0000000000000000)</td>
<td>0(x0000000000000000)</td>
<td>0(x0000000000000000)</td>
</tr>
<tr>
<td>10-round related-key differential trail with probability $2^{-55}$</td>
<td>0 (0x0000000000000000)</td>
<td>0(x0000000000000000)</td>
<td>0(x0000000000000000)</td>
</tr>
</tbody>
</table>

Table 7: S-box $S$.

<table>
<thead>
<tr>
<th>X</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>S[X]</td>
<td>c</td>
<td>4</td>
<td>b</td>
<td>1</td>
<td>7</td>
<td>9</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>2</td>
<td>0</td>
<td>f</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>
(v) before_sbox_difference_i,j =
before_sbox.value1_i,j @ before_sbox.value2_i,j
(vi) after_sbox_difference_i,j =
after_sbox.value1_i,j & after_sbox.value2_i,j
(vii) before_rotation_difference_i,j: difference of the
jth block before bit rotation of the ith round
(viii) after_rotation_difference_i,j: difference of the jth
block after bit rotation of the ith round
(ix) after_blockxor_difference_i,j: difference of the jth
block after block XOR operation of the ith round
(x) flag_i,j: represents whether the jth S-box in the ith round
is active or not
(xi) total_num: the total number of active S-boxes in the trail
(xii) probability_i,j: probability parameter of the jth S-
box in the ith round
(xiii) total_probability: probability parameter of the
whole differential trail

3.1.1. Permutation Layer. The Permutation Layer is made up
of three linear operations. Constraints about these linear
operations are listed in Algorithm 1, 2, and 3, respectively.

3.1.2. Substitution Layer. Constraints built in Algorithm 4
describe difference propagations through S-boxes, which
include initializing variables by values of S-boxes and
showing relationships among variables.

Other constraints should be added in Algorithm 4, when
searching for differential trails with the minimal number
of active S-boxes. These constraints of searching for the
minimal number of active S-boxes are illustrated in Algo-

Figure 1: Round function.

Algorithm 4. The variable flag_i,j is used to explain whether an S-
box is active or not. When the jth S-box in the ith round is
active, flag_i,j = 1; otherwise, flag_i,j = 0.

We need to check whether total_num = n is satisfied or
not under constraints from Algorithm 4 and 5, where n is the
expected number of active S-boxes in the whole differential
trail, and total_num = n is regarded as the objective

We first search for 3-round trails with the minimal
number of active S-boxes, and set n = 1. Put all constraints
including Algorithm 1–5 into STP and check the satisfaction
of these objective constraints. If STP returns invalid,
gradually adjust the value of n by adding 1 till STP returns
a trail. When searching for r-round trails with the minimal
number of active S-boxes, we set the value of n equal to the
minimal number of active S-boxes of (r − 1)-round trails.
Repeat the above steps.

When searching for the optimal differential trail, we
need to consider probabilities of difference propagations
through S-boxes. Probabilities of S-boxes could be described
by the difference distribution table (DDT).

For any input difference x and output difference y of the
S-box, the probability of the S-box, named p_sbox(x, y), is
equal to the corresponding value in the DDT divided by 16,
which is

\[ p_{sbox}(x, y) = \frac{\text{value_in_DDT}(x, y)}{16} \]  

We can write that \( p_{sbox}(x, y) = 2^{-\text{probability}(x, y)} \). In our
model, we use the parameter probability(x, y) to present the

\[ \text{probability}(x, y) = -\log_2 \frac{\text{value_in_DDT}(x, y)}{16} \]  

Values of probability(x, y) for all x and y are stored in
a table called Diff_dist.

Denote the probability of the whole differential trail as

\[ 2^{-\text{total_probability}} \]  

and

\[ \text{total_probability} = \sum_{i=0, j=0}^{i=15, j=15} \text{probability}_i,j. \]  

The parameter total_probability is used to represent the
probability of the whole trail in the constraints. Constraints
of finding the optimal differential trail are presented in
Algorithm 6.

Let that the value of total_probability starts from dou-
bling the minimal number of active S-boxes because
consider probabilities of all active S-boxes in the trail are optimal. Adjust the value of total probability by adding 1 till obtaining the trail with the optimal probability.

3.2. Finding Linear Trails. Due to differences between difference propagations and linear mask propagations, we need to build constraints suitable for tracking linear masks in all operations of BORON. Notations are given as follows:

(i) before_sbox_mask_{i,j}: input mask of the jth S-box in the ith round

(ii) after_sbox_mask_{i,j}: output mask of the jth S-box in the ith round

(iii) branch_{i,j}_up, branch_{i,j}_down, branch_{i,j}_another: masks of the jth branch operation in the block XOR operation of the i-th round

(iv) xor_{i,j}_up, xor_{i,j}_down, xor_{i,j}_another: masks of the jth xor operation in the block XOR operation of the ith round

(v) bias_{i,j}: bias parameter of the jth S-box in the ith round

\[ r: \text{the number of rounds} \]
\[
\text{for } i \leftarrow 0 \text{ to } r \text{ do} \\
\text{before_rotation_difference}_{i,0} = \\
\text{after_rotation_difference}_{i,1} \oplus \text{before_rotation_difference}_{i,0} \ll 1 \\
\text{after_rotation_difference}_{i,2} = \text{before_rotation_difference}_{i,1} \ll 4 \\
\text{after_rotation_difference}_{i,3} = \text{before_rotation_difference}_{i,2} \ll 7 \\
\text{after_rotation_difference}_{i,0} = \text{after_rotation_difference}_{i,0} \ll 9 \\
\end{for}

Algorithm 1: Difference propagations through nibble shift.

\[ r: \text{the number of rounds} \]
\[
\text{for } i \leftarrow 0 \text{ to } r \text{ do} \\
\text{blockxor_difference}_{i,1} = \\
\text{rotation_difference}_{i,1} \oplus \text{after_rotation_difference}_{i,3} \\
\text{blockxor_difference}_{i,2} = \text{rotation_difference}_{i,0} \oplus \text{after_rotation_difference}_{i,2} \\
\text{blockxor_difference}_{i,0} = \text{blockxor_difference}_{i,0} \oplus \text{after_rotation_difference}_{i,1} \\
\text{blockxor_difference}_{i,3} = \text{rotation_difference}_{i,3} \oplus \text{after_rotation_difference}_{i,2} \\
\end{for}

Algorithm 2: Difference propagations through bit rotation.

\[ r: \text{the number of rounds} \]
\[
\text{for } i \leftarrow 0 \text{ to } r \text{ do} \\
\text{blockxor_difference}_{i,1} = \\
\text{rotation_difference}_{i,1} \oplus \text{after_rotation_difference}_{i,3} \\
\text{blockxor_difference}_{i,2} = \text{rotation_difference}_{i,0} \oplus \text{after_rotation_difference}_{i,2} \\
\text{blockxor_difference}_{i,0} = \text{blockxor_difference}_{i,0} \oplus \text{after_rotation_difference}_{i,1} \\
\text{blockxor_difference}_{i,3} = \text{rotation_difference}_{i,3} \oplus \text{after_rotation_difference}_{i,2} \\
\end{for}

Algorithm 3: Difference propagations through block XOR operation.
3.2.1. Permutation Layer. Situations that linear masks propagate nibble shift and bit rotation are the same as situations that differences do. So, we omit constraints about tracking linear masks through nibble shift and bit rotation. Refer Algorithm 1 and 2 for details.

It is little complicated that linear masks propagate through block XOR operation due to the property of linear masks. Detailed constraints are described in Algorithm 7. Figure 2 illustrates the variables which are used in tracking linear masks through block XOR operation.

3.2.2. Substitution Layer. Similar to finding differential trails, we also use flag\_i\_j to represent whether an S-box is active or not when finding linear trails with the minimal number of active S-boxes. And, the objective constraint is total_num = n, where n is the expected number of active S-boxes in the whole linear trail. The choice of n is the same as above. Detailed constraints are illustrated in Algorithm 8.

When searching for the optimal linear trail, we need to consider biases of S-boxes which could be described by the linear approximation table (LAT).

For any input mask \( \alpha \) and output mask \( \beta \) of the S-box, the bias of the S-box, named \( b_{sbox}(\alpha, \beta) \) is equal to the corresponding value in the LAT divided by 16, which is

\[
b_{sbox}(\alpha, \beta) = \frac{\text{value in LAT}(\alpha, \beta)}{16}.
\]

We can write that \( b_{sbox}(\alpha, \beta) = 2^{\text{bias}(\alpha, \beta)} \).

Use the parameter bias\( (\alpha, \beta) \) to present the bias of the S-box in the constraints and

\[
\text{Algorithm 4: Difference propagations through S-boxes.}
\]

\[
\text{Algorithm 5: Finding differential trails with the minimal number of active S-boxes.}
\]
bias(\(\alpha, \beta\)) = -\log_{2} \left( \frac{\text{value in \text{LAT}(\(\alpha, \beta\))}}{16} \right) \tag{5} \]

Denote the bias of the whole linear trail as \(2^{-\text{total bias}}\) and due to the Piling-up Lemma [10], we have

\[
\text{total bias} = 1 - \text{total num} + \sum_{i=0, j=0}^{i \leq r, j \leq 15} \text{bias}_{i, j} \tag{6}
\]

Values of bias(\(\alpha, \beta\)) for all \(\alpha\) and \(\beta\) are stored in a table named linear_dist.
Constraints of finding linear trails with the optimal bias are presented in Algorithm 9.

Set the initial value of total_bias by considering all S-boxes with the optimal bias. Adjust the value of total_bias by adding 1 till obtaining the trail with the optimal bias.

3.3. Finding Related-Key Differential Trails. When searching for differential trails in related-key setting, some constraints are given in Algorithm 10 to illustrate the related-key relationship. And we also need to construct some constraints which describe difference propagations through the key schedule. Operations in the key schedule are similar to those in the encryption algorithm and constraints could refer to Algorithm 2, 3, and 6.

Detailed codes are presented in the GitHub and the optimal trails are illustrated in Table 6.

3.4. Finding Impossible Differential Trails. In order to find impossible differential trails, we need to set the pattern of input and output differences and check the satisfaction of constraints. If the solver returns a trail, there is no impossible differential trail under this pattern. If returns invalid, an impossible differential trails are found. Constraints are listed in Algorithm 11.

As a result, we obtain 4 different 7-round impossible differential trails, which are

\[
\begin{align*}
0x0000000000010000 & \rightarrow 0x0001000000000000, \\
0x0000000000010000 & \rightarrow 0x0010000000000000, \\
0x0000000000100000 & \rightarrow 0x0100000000000000, \\
0x0000000000100000 & \rightarrow 0x1000000000000000.
\end{align*}
\]

The former is the input pattern, and the latter is the output pattern of the 7-round BORON, where “1” represents the active nibble and “0” represents the inactive nibble.

4. Key-Recovery Attacks

4.1. Differential Cryptanalysis. Considering the 8-round differential trail with the probability of $2^{-62}$ as a distinguisher illustrated in Table 4, a key-recovery attack towards 9-round BORON is presented in the following by adding one more round on the tail of the distinguisher. We omit the permutation layer of the last round due to linearity. We adopt the 80-bit key schedule in this key-recovery attack.

The input difference of this distinguisher is $0x0000080000100000$, and the output difference is $0x0041004100080009$. We choose $2^{62}$ plaintext pairs whose differences are equal to $0x0000080000100000$. Filter these plaintext pairs and keep ones whose ciphertext differences are equal to $0x00??00??00??00??$, where ? represents the
unknown difference. On average, $2^{22}$ plaintext pairs are left because the probability that a pair satisfies the limitation of this distinguisher is $2^{-40}$.

Guess 24 bits of subkey $K_y$ including $k_0$, $k_1$, $k_2$, $k_3$, $k_{16}$, $k_{17}$, $k_{18}$, $k_{19}$, $k_{32}$, $k_{33}$, $k_{34}$, $k_{35}$, $k_{36}$, $k_{37}$, $k_{38}$, $k_{39}$, $k_{48}$, $k_{49}$, $k_{50}$, $k_{51}$, $k_{52}$, $k_{53}$, $k_{54}$, and $k_{55}$. Set $2^{24}$ counters and initialize their

\[ r: \text{the number of rounds} \]
\[ 2^{-b}: \text{the expected bias of the whole trail} \]
\[ (1) \text{for } a \leftarrow 0 \text{ to } 15 \text{ do} \]
\[ (2) \text{for } \beta \leftarrow 0 \text{ to } 15 \text{ do} \]
\[ (3) \quad \text{linear_dist}[a][\beta] = \text{bias}(a, \beta) \]
\[ (4) \text{end for} \]
\[ (5) \text{end for} \]
\[ (6) \text{for } i \leftarrow 0 \text{ to } r \text{ do} \]
\[ (7) \text{for } j \leftarrow 0 \text{ to } 15 \text{ do} \]
\[ (8) \quad \text{bias}_{i,j} = \text{linear_dist}[\text{before_sbox_mask}_{i,j}][\text{after_sbox_mask}_{i,j}] \]
\[ (9) \text{end for} \]
\[ (10) \text{end for} \]
\[ (11) \text{for } i \leftarrow 0 \text{ to } r \text{ do} \]
\[ (12) \quad \text{bias}_{i} = \text{linear_dist}_{\text{before_sbox_mask}_{i}}[\text{after_sbox_mask}_{i}] \]
\[ (13) \text{end for} \]
\[ (14) \text{total_bias} = 1 - \text{total_num} + \sum_{i=0, j=0}^{r-1, j=15} \text{bias}_{j} \]
\[ (15) \text{total_bias} = b \]

Algorithm 9: Finding linear trails with the optimal bias.

\[ r: \text{the number of rounds} \]
\[ x: \text{input pattern of the trail} \]
\[ y: \text{output pattern of the trail} \]
\[ (1) \text{flag.head} = \text{flag.0.0} \]
\[ (2) \text{for } i \leftarrow 1 \text{ to } r \text{ do} \]
\[ (3) \quad \text{flag.head} = \text{flag.0.i} \|	ext{flag.head} \]
\[ (4) \text{end for} \]
\[ (5) \text{flag.tail} = \text{flag.}(r-1).0 \]
\[ (6) \text{for } i \leftarrow 1 \text{ to } 15 \text{ do} \]
\[ (7) \quad \text{flag.tail} = \text{flag.}(r-1).i \|	ext{flag.tail} \]
\[ (8) \text{end for} \]
\[ (9) \text{flag.head} = x \]
\[ (10) \text{flag.tail} = y \]

values as zero. Each candidate subkey has a counter. And, values of counters represent the number of plaintext pairs that satisfy the distinguisher under the corresponding candidate subkey.

For the right candidate subkey, one plaintext pair satisfies the distinguisher. However, for wrong candidate subkeys, $2^{-24}$ plaintext pair satisfies the distinguisher on average. We pick the candidate subkey whose counter is maximum as the right subkey. The time complexity is $2^{46}$, the memory complexity is $2^{24}$, and the data complexity is $2^{63}$.

For the 80-bit master key, we have already guessed 24 bits. To obtain the left 56 bits, we use a brute-force search. The time complexity is $2^{56}$. The memory and data complexity could be omitted.

Hence, we give a key-recovery attack towards 9-round BORON with time complexity $2^{56}$, memory complexity $2^{24}$, and data complexity $2^{63}$.

We use the method in [21] to evaluate the success probability of this attack. The success probability $P_s$ is computed as follows:

$$P_s = \Phi \left( \frac{\sqrt{\mu S_N} - \Phi^{-1} (1 - 2^{-a})}{\sqrt{S_N + 1}} \right),$$  \hspace{1cm} (8)

where $\mu = p N$, $p$ is the probability of the differential trail, $N$ is the number of plaintext pairs, $a$ is the advantage, and $S_N$ is the signal-to-noise ratio. In our differential cryptanalysis,

$$P_s = \Phi \left( \frac{\sqrt{2^{-62} \times 2^{62} \times 2^{34}} - \Phi^{-1} (1 - 2^{-1})}{\sqrt{2^{34} + 1}} \right) = 0.841.  \hspace{1cm} (9)$$

### 4.2. Linear Cryptanalysis

Based on the 9-round optimal linear trail listed in Table 5, we could present a key-recovery attack against 11-round BORON by linking one more round at the head and tail of this 9-round linear trail.

The number of plaintexts required in the key-recovery attack is equal to $c \epsilon^2$, where $\epsilon$ is the bias of the linear approximation and $c$ is a constant [10]. Set $c = 8$, and then we need $2^{63}$ plaintexts to achieve this attack and the success probability of this attack is $96.7\%$.

We need to guess 28 bits of subkey $K_0$ and 24 bits of subkey $K_{11}$. Considering the 128-bit key schedule, there are 10 same bits between $K_0$ and $K_{11}$. In total, we need to guess 42 bits to obtain 5-bit key information. This key-recovery attack requires $2^{63} \times 2^{42} = 2^{105}$ one-round encryptions, which is equivalent to $2^{101.54}$ 11-round encryptions. The left 123 bits need to be searched by brute-force.

Hence, the time complexity is $2^{123}$, memory complexity is $2^{42}$, and data complexity is $2^{63}$.

### 5. Conclusion

In this paper, we present the first third-party cryptanalysis of the lightweight block cipher BORON against differential and linear cryptanalysis. By the automatic tool, we search for differential and linear trails with the minimal number of active S-boxes and trails with the optimal probability and bias. Considering the optimal trails as distinguishers, we mount key-recovery attacks. Utilizing the 8-round differential trail with the optimal probability $2^{-62}$, we give a key-recovery attack towards 9-round BORON whose time complexity is $2^{56}$, data complexity is $2^{63}$, and memory complexity is $2^{24}$. By the 9-round linear trail with the optimal bias $2^{-30}$, we describe the key-recovery attack towards 11-round BORON whose time complexity is $2^{123}$, data complexity is $2^{63}$, and memory complexity is $2^{42}$. Besides differential and linear cryptanalysis, there are other powerful cryptanalysis techniques. Further security evaluations could be made in future work.

### Data Availability

The codes and trails used to support the findings of this study have been deposited in the GitHub (https://github.com/CatherineLiang/Cryptanalysis-of-BORON).

### Conflicts of Interest

The authors declare that they have no conflicts of interest.

### Acknowledgments

The authors thank Associate Professor Wu Hongjun from Nanyang Technological University for careful discussion. This work was supported by the National Cryptography Development Fund (no. MMJ20170102), National Natural Science Foundation of China (grant no. 61572293), Major Scientific and Technological Innovation Projects of Shandong Province, China (no. 2017CXGC0704), Fundamental Research Fund of Shandong Academy of Sciences (grant no. 201812-16), and the China Scholarship Council (no. CSC201906220073).

### References


