Theoretical Linear Cryptanalysis of the 5G Standard Candidate SNOW 5G

Yinuo Liu, Jing Yang and Tian Tian

Information Engineering University, Zhengzhou, China
liuyinuo_terry@163.com, yangjingfi@163.com, tiantian_d@126.com

Abstract. In this paper, we perform linear cryptanalysis of the stream cipher SNOW 5G, which is recommended by the international standardization group (SAGE) as one standard algorithm for 5G confidentiality and integrity protection over the wireless channel. SNOW 5G can be regarded as one member of the SNOW-V family, as it is modified from SNOW-Vi by SAGE with a slight improvement. As an overall contribution, we provide a comprehensive and elaborate theoretical analysis of linear approximations of SNOW 5G and provide the best public cryptanalysis result by far. Specifically, we first theoretically analyze the formats of linear masks of SNOW 5G that can introduce high correlations, and then search for high-quality linear masks using a divide-and-conquer method based on the different cases of a critical intermediate linear mask. We find a linear approximation of SNOW 5G with correlation $-2^{-67.67}$ and further launch a correlation attack against it with complexity $2^{279.8}$, improving the existing best correlation attack by a factor of $2^{32.4}$. Our results are mainly from theoretical analysis, which involve little computation overhead and help to better understand the security of SNOW 5G.

Keywords: SNOW 5G · linear cryptanalysis · linear approximations · divide-and-conquer

1 Introduction

The SNOW series of stream ciphers have played an essential part in the confidentiality and integrity protection in various generations of mobile communication systems. SNOW 3G is standardized in 3G and 4G, providing 128-bit security levels, and is usually implemented in hardware to provide high speeds. When it comes to 5G, the fundamental changes in the system architecture and new demands in security pose new requirements for the cryptographic algorithms [3rd]. Basically, these ciphers need to provide a speed of at least 20 Gbps (the peak data rate of downlink transmission in 5G) in virtualized environments and a security level of 256 bits. SNOW 3G is inadequate for 5G as current benchmarks give approximately 9 Gbps in software [YJ20], and more importantly, it is shown to be susceptible to linear cryptanalysis [YJM19]. In 2018, the international mobile broadband standard organization 3GPP asked SAGE (Security Algorithms Group of Experts) to select and evaluate new cryptographic primitives for 5G.

As a response, the SNOW-V cipher was proposed in 2019 by researchers from Lund University and Ericsson [EJMY19] targeted to be used in 5G, which achieves rates up to 58 Gbps and provides 256 bits of security. The designers published an extreme performance variant called SNOW-Vi the next year to further increase the speeds in software environments. The ciphers were submitted to SAGE and received wide attention. Driven by the linear cryptanalysis results [SJZ+22] of SNOW-V and SNOW-Vi in 2022, SAGE proposed to replace the 32-bit modular additions in SNOW-Vi with 16-bit ones, and the new variant is called SNOW 5G [lia11, Tea, 3GP03]. SNOW 5G is recommended...
by SAGE as one standard algorithm for 5G confidentiality and integrity protection over the air, but we note that the desired specifications are not ready because they are still under construction [3GP03].

Till now, the SNOW-V series have received much public evaluation, including guess-and-determine attacks [JLH20, YJM21, JHL23], linear cryptanalysis [GZ21, YJM21, ZFZ22, SJZ+22], integral attack [HII+22], and differential attack [HII+22], etc. Below, we give a more detailed introduction to those guess-and-determine attacks and linear cryptanalysis.

In [JLH20], Jiao et al. proposed a guess-and-determine attack against SNOW-V with a time complexity of $2^{406}$, using only seven keystream words. The guess-and-determine attack was improved in [YJM21] with a complexity of $2^{378}$ by carefully designing the order of guessing and truncating as many invalid guessing paths as possible. Jiao et al. further analyzed that the techniques in [YJM21] do not apply to SNOW-Vi, and proposed a guess-and-determine attack against it with complexity $2^{408}$ in [JHL23].

For linear cryptanalysis, Gong and Zhang investigated the linear approximations of SNOW-V in [GZ21] and proposed correlation attacks against its three simplified variants. For the closest variant SNOW-V$_{16,16}$, in which one 32-bit modular addition is replaced with four byte-based additions, they mounted a first correlation attack with time, memory, and data complexities of $2^{377.01}$, $2^{363}$, and $2^{553}$, respectively. Yang et al. in [YJM21] also considered a reduced variant SNOW-V$_{50}$ where modular additions are replaced with XOR operations, and demonstrated a distinguishing attack with complexity $2^{303}$. All of the attacks above have complexities higher than that of exhaustive key search.

The outstanding cryptanalysis of SNOW-V/Vi is from the work [SJZ+22] presented at Eurocrypt 2022, in which correlation attacks with complexities less than exhaustive key search were proposed when ignoring the constraint of the length of the keystream under one pair of key and IV. Inspired by the linear cryptanalysis techniques in block ciphers, the authors converted the linear approximation involving three consecutive keystream words into an equivalent composite function. By modeling the correlation propagation between sub-functions and using automatic search tools based on SAT/SMT, they searched out a series of linear trails with high correlations. They derived a binary linear approximation with correlation $-2^{-47.76}$. When ignoring the constraint of the keystream length, they can launch a correlation attack with time, memory, and data complexities of $2^{246.53}$, $2^{238.77}$, and $2^{237.5}$, respectively. They later extended the techniques to other word-based stream ciphers and proposed a general correlation evaluation model for them [SJZ+23].

Zhou et al. [ZFZ22] also proposed two algorithms to efficiently evaluate linear approximations of SNOW-V with linear hull effect taken into consideration. They built multiple pre-computed linear approximation tables (LATs) corresponding to different input-output linear masks for the main components used in the finite state machine (FSM) in SNOW-V/Vi, and used them to choose high-quality linear masks during the online phase. The best linear approximation they derived gave an absolute correlation $2^{-47.567}$, which led to a fast correlation attack against SNOW-V with time, memory, and data complexities $2^{240.86}$, $2^{240.37}$, and $2^{236.87}$, respectively.

Till now, there have been very few public cryptanalysis results of SNOW 5G. As far as we know, the only cryptanalysis is from [MJS+23] which applies the techniques in [SJZ+22] to search for linear trails using automatic searching tools. A linear approximation with correlation $2^{-75.82}$ is found and used to launch a correlation attack with complexity $2^{312.24}$.

As a candidate standard for 5G confidentiality and integrity protection, SNOW 5G still needs more comprehensive cryptanalysis to increase our confidence in its security. In this paper, we aim to investigate the linear cryptanalysis of SNOW 5G.

**Contributions.** The fundamental novelty and contribution of this paper is the theoretical analysis of the linear approximations of SNOW 5G. The linear masks for SNOW 5G, as well as the other SNOW-V variants, derived in [SJZ+22, YJM21, ZFZ22, SJZ+23, MJS+23] were mainly obtained from automatic searching tools or multiple linear approximation
Table 1: Attacks against the SNOW-V series ($T/D/M$ denote the time, data, and memory complexities, respectively)

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Target</th>
<th>Complexities $T/D/M$</th>
<th>Techniques</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>GnD Attacks</td>
<td>SNOW-V</td>
<td>$2^{306}$/896/N/A</td>
<td>auto-searching</td>
<td>[JLH20]</td>
</tr>
<tr>
<td></td>
<td>SNOW-V</td>
<td>$2^{384}$/896/N/A</td>
<td>truncating</td>
<td>[YJM21]</td>
</tr>
<tr>
<td></td>
<td>SNOW-V</td>
<td>$2^{378}$/1024/$2^{128}$</td>
<td>truncating</td>
<td>[YJM21]</td>
</tr>
<tr>
<td></td>
<td>SNOW-Vi</td>
<td>$2^{408}$/896/N/A</td>
<td>auto-searching</td>
<td>[JHL23]</td>
</tr>
<tr>
<td>Linear Attacks</td>
<td>SNOW-V</td>
<td>$2^{377}$/2/254/$2^{363}$</td>
<td>LAT</td>
<td>[GZ21]</td>
</tr>
<tr>
<td></td>
<td>SNOW-V/Vi</td>
<td>$2^{303}$/2/303</td>
<td>LAT</td>
<td>[YJM21]</td>
</tr>
<tr>
<td></td>
<td>SNOW-V</td>
<td>$2^{246.5}$/2/237.5/$2^{238.8}$</td>
<td>SAT/SMT</td>
<td>[SZ22]</td>
</tr>
<tr>
<td></td>
<td>SNOW-V</td>
<td>$2^{240.9}$/2/236.9/$2^{240.4}$</td>
<td>LAT</td>
<td>[ZJ22]</td>
</tr>
<tr>
<td></td>
<td>SNOW 5G</td>
<td>$2^{412.2}$/2/297.1/$2^{297.1}$</td>
<td>MILP</td>
<td>[MJS23]</td>
</tr>
<tr>
<td></td>
<td>SNOW 5G</td>
<td>$2^{179.8}$/2/265.4/$2^{266.5}$</td>
<td>theoretical analysis</td>
<td>This paper</td>
</tr>
<tr>
<td>Integral Attack</td>
<td>SNOW-V</td>
<td>$2^{48}$/2/2/279</td>
<td>MILP**</td>
<td>[HII22]</td>
</tr>
<tr>
<td>Differential Attack</td>
<td>SNOW-V</td>
<td>$2^{154}$/2/27/N/A **</td>
<td>MILP</td>
<td>[HII22]</td>
</tr>
</tbody>
</table>

* Five initialization rounds attacked. ** Four initialization rounds attacked. *** MILP: mixed-integer linear programming.

While the theoretical analysis is still needed to help understand the security of the ciphers in depth. In our method, we theoretically analyze the linear approximations of small components and then derive connections and constraints between different linear masks. After that, we use a divide-and-conquer method to search for high-quality linear masks according to the different cases of a critical intermediate linear mask. Such a search process relies heavily on theoretical analysis and requires little computation overhead. We find a linear trail with correlation $2^{-67.67}$ and experimentally verify that it is a dominant trail under the outer linear masks we find. Specifically, we fix the outer linear masks and compute the correlations under other possible intermediate linear masks. The linear trail exhibits a significantly higher correlation than the others, which can represent the total correlation of the linear approximation with a negligible error. Therefore, the linear approximation has a correlation of $2^{-67.67}$, improving the existing benchmark $2^{-75.82}$ in [MJS23] by $2^{8.15}$. We further launch a correlation attack against SNOW 5G with complexity $2^{279.8}$, improving the existing best attack in [MJS23] by a factor of $2^{32.4}$. The complexity is still higher than exhaustive key search, but the results demonstrate the effectiveness of the method. The theoretical analysis provides much help in understanding the structures of linear approximations of SNOW 5G and its resistance against linear cryptanalysis. Table 1 lists the main cryptanalysis results against the SNOW-V series and the comparison with the new results in this paper.

Organization. The paper is organized as follows. We first present some basic notations, definitions, and a brief introduction to SNOW 5G in Section 2. In Section 3, we first present a theoretical analysis of the linear approximations and then apply it to search for high-quality linear masks of SNOW 5G. After that, we present the results and propose a correlation attack against SNOW 5G in this section as well. We draw a conclusion in Section 4.

2 Preliminaries

2.1 Notations and Definitions

We below present some notations and definitions.

$\mathbb{F}_2$ the binary field
$F_2^n$ the $n$-dimension vector space over $F_2$
$F_{2^n}$ the finite field with $2^n$ elements
$\oplus$ the bitwise XOR operation
$\oplus_m$ the addition modulo $2^n$
$E(\cdot)$ the encryption round function of AES, with roundkey $= 0$
$E^{-1}(\cdot)$ the decryption round function of AES, with roundkey $= 0$
$\Pr(A)$ the probability of the event $A$
$w_H(x)$ the Hamming weight of a Boolean vector $x$

By referring to the AES state representation, we express all the 128-bit internal states and linear masks in the form of $4 \times 4$ byte matrices. As an example, $S = (w_{15}, w_{14}, \ldots, w_0)$, where $w_i (15 \geq i \geq 0)$ are bytes, can be denoted as

$$
\begin{bmatrix}
w_0 & w_4 & w_8 & w_{12} \\
w_1 & w_5 & w_9 & w_{13} \\
w_2 & w_6 & w_{10} & w_{14} \\
w_3 & w_7 & w_{11} & w_{15}
\end{bmatrix}
$$

Throughout the paper, we use rows and columns in the matrix representation to index the byte positions of the states and linear masks. The index range for both rows and columns is from 0 to 3. For example, column 1 of $S$ involves bytes 4 to 7 ($w_4, w_5, w_6, w_7$), which also indicates the second 32-bit word of $S$; while row 2 of $S$ involves bytes $w_2, w_6, w_{10},$ and $w_{14}$.

**Definition 1.** Given two $n$-bit binary vectors $a = (a_{n-1}, a_{n-2}, \ldots, a_0)$ and $b = (b_{n-1}, b_{n-2}, \ldots, b_0)$, the inner product is computed as $a \cdot b = \sum_{i=0}^{n-1} a_i \cdot b_i$.

**Definition 2.** Given a binary random variable $x$, the correlation of $x$ is defined as $c(x) = \Pr(x = 0) - \Pr(x = 1)$. Given a Boolean expression in $n$ binary variables $f(x_0, x_1, \ldots, x_{n-1})$, the correlation of $f$ is defined as $c(f) = \Pr(f = 0) - \Pr(f = 1)$.

**Definition 3.** Given an $n$-bit binary vector $a$ and a factor $m$ of $n$, when applying parallel $\oplus_m$ operations to the vector $a$, $a$ can be divided into $n/m$ blocks and each block is composed of $m$ bits called one $\oplus_m$-block.

In our analysis, we have $n = 128, m = 16$. We also use $\oplus_{16}$ to denote 8 parallel 16-bit modular additions without ambiguity.

**Definition 4.** A variable is called isolated if it only appears in the linear part of the algebraic normal form (ANF) of a Boolean expression.

For example, $x_0$ is an isolated variable in $f(x_0, x_1, \ldots, x_{n-1}) = f_1(x_1, x_2, \ldots, x_{n-1}) \oplus x_0$.

**Definition 5.** Given two $n$-bit vectors $a = (a_{n-1}, a_{n-2}, \ldots, a_0)$ and $b = (b_{n-1}, b_{n-2}, \ldots, b_0)$, we say that $a$ and $b$ have the same $(n-i)$-bit zero suffix if $a_{n-1} = a_{n-2} = \cdots = a_i = b_{n-1} = b_{n-2} = \cdots = b_i = 0$ and $a_{i-1} = b_{i-1} = 1$.

Assuming that $x_1, x_2, \ldots, x_k$ are $k$ independent binary random variables with correlations $c_1, c_2, \ldots, c_k$, respectively, the correlation of $x_1 \oplus x_2 \oplus \cdots \oplus x_k$ can be computed by $c = \prod_{j=1}^{k} c_j$ according to the well-known Piling-up Lemma [Mat93].

### 2.2 Brief Introduction to SNOW 5G

The SNOW-V series of ciphers inherit from the predecessor SNOW 3G that adopt LFSRs as the source of pseudorandomness and FSMs to disrupt the linearity. However, the registers are increased to larger sizes and better aligned to accommodate the advanced
intrinsic instructions to achieve high speeds when implemented in software environments. For more design details about SNOW-V, SNOW-Vi, and SNOW 5G, we refer to [EJMY19, EMJY21, lia11]. Below, we give an introduction to SNOW 5G.

Figure 1 illustrates the schematic of SNOW 5G, which consists of a circular construction of two LFSRs and one FSM accommodating three 128-bit registers. The LFSR part is built over two LFSRs, called LFSR-A and LFSR-B, each with 16 stages denoted as \((a_{15}, \ldots, a_1, a_0)\) and \((b_{15}, \ldots, b_1, b_0)\), respectively. Each stage of LFSR-A and LFSR-B holds 16 bits, which are generated by polynomials \(g^A(x)\) and \(g^B(x)\), respectively, defined as

\[
g^A(x) = x^{16} + x^{14} + x^{11} + x^9 + x^6 + x^5 + x^3 + x^2 + 1 \in \mathbb{F}_2[x],
\]

\[
g^B(x) = x^{16} + x^{15} + x^{14} + x^{11} + x^{10} + x^7 + x^2 + x + 1 \in \mathbb{F}_2[x].
\]

We denote \(\alpha \in \mathbb{F}_{2^{16}}^A\) and \(\beta \in \mathbb{F}_{2^{16}}^B\) to represent the roots of \(g^A(x)\) and \(g^B(x)\), respectively. In each clock, \(a_0, b_0\) are removed from the LFSRs, \(a_i\) and \(b_i\) \((1 \leq i \leq 15)\) are shifted to \(a_{i-1}\) and \(b_{i-1}\), respectively, and the values in \(a_{15}\) and \(b_{15}\) are newly updated by

\[
a_{15}^{(t+1)} = b_0^{(t)} + \alpha a_0^{(t)} + a_7^{(t)} \mod g^A(\alpha),
\]

\[
b_{15}^{(t+1)} = a_0^{(t)} + \beta b_0^{(t)} + b_8^{(t)} \mod g^B(\beta).
\]

Each time the LFSRs update, they will update eight times, meaning that the high eight stages (256 bits in total) of LFSR-A and LFSR-B are updated. After that, two 128-bit taps \(T_1\) and \(T_2\), represented below, are extracted and fed to the FSM:

\[
T_1^{(t)} = (b_{15}^{(8t)}, b_{14}^{(8t)}, \ldots, b_8^{(8t)}),
\]

\[
T_2^{(t)} = (a_{15}^{(8t)}, a_{14}^{(8t)}, \ldots, a_8^{(8t)}).
\]
The FSM part includes three 128-bit registers $R_1, R_2, R_3$ and takes $T_1$ and $T_2$ from the LFSRs to produce one 128-bit keystream word and update the registers. The keystream word is generated by

$$z^{(t)} = (R_1^{(t)} \oplus_{16} T_1^{(t)}) \oplus R_2^{(t)},$$

(1)

where $\oplus_{16}$ denotes eight parallel 16-bit modular additions. It is the operation modified from the 32-bit modular addition in SNOW-V/Vi.

The registers are then updated by

$$R_1^{(t+1)} = \sigma(R_2^{(t)} \oplus_{16} (R_3^{(t)} \oplus T_2^{(t)})),$$

$$R_2^{(t+1)} = E(R_1^{(t)}),$$

$$R_3^{(t+1)} = E(R_2^{(t)}),$$

where $\sigma$ is a byte-oriented permutation defined as

$$\sigma = [0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15],$$

which means that byte 0 is moved to position 0, byte 4 is moved to position 1, byte 8 is moved to position 2, and so on. If we use the matrix representation, then applying the $\sigma$ operation to a 128-bit variable $S$ leads to

$$\sigma(S) = \begin{bmatrix}
    w_0 & w_1 & w_2 & w_3 \\
    w_4 & w_5 & w_6 & w_7 \\
    w_8 & w_9 & w_{10} & w_{11} \\
    w_{12} & w_{13} & w_{14} & w_{15}
\end{bmatrix}.$$

During the initialization phase, a 256-bit key and a 128-bit IV are loaded into the LFSRs, and the registers in the FSM are initialized to zero. The initialization consists of 16 steps, during which the cipher is updated in the same way as that in the running-key mode, but with the output from the FSM feeding back to the LFSRs instead of producing a keystream word. Additionally, at the last two steps of the initialization phase, we XOR the key to the $R_1$ register. Since the attack in this paper only involves the keystream mode, we do not give more details about the initialization mode but refer to [EJMY19].

### 3 Linear Cryptanalysis of SNOW 5G

In this section, we first give the linear approximation of SNOW 5G and divide the approximation noises into several parts in Section 3.1. We then analyze the linear approximations of the underlying non-linear operations and derive the patterns of high-quality linear masks in Section 3.2. After that, we use a divide-and-conquer method to search for linear masks based on the different cases of a critical intermediate linear mask in Section 3.3. We present the results in Section 3.4 and propose a correlation attack against SNOW 5G in Section 3.5.

#### 3.1 Linear Approximations and Division of Noises

We denote $R_1^{(t)}$, $R_2^{(t)}$, and $R_3^{(t)}$ as $R_1$, $R_2$, and $R_3$, respectively, for ease of notation. The keystream words produced in three consecutive clocks can be expressed as

$$z^{(t-1)} = (E^{-1}(R_2) \oplus_{16} T_1^{(t-1)}) \oplus E^{-1}(R_3),$$

$$z^{(t)} = (R_1 \oplus_{16} T_1^{(t)}) \oplus R_2,$$

$$z^{(t+1)} = (\sigma(R_2 \oplus_{16} (R_3 \oplus T_2^{(t)})) \oplus_{16} T_1^{(t+1)}) \oplus E(R_1).$$
We apply three 128-bit linear masks \( \alpha, \beta, \) and \( \gamma \) to \( z^{(t-1)}, z^{(t)}, \) and \( z^{(t+1)} \), respectively, and get
\[
\begin{align*}
\alpha \cdot z^{(t-1)} &= \alpha \cdot (E^{-1}(R_2) \boxplus_1 T_1^{(t-1)}) \oplus \alpha \cdot E^{-1}(R_3), \\
\beta \cdot z^{(t)} &= \beta \cdot (R_1 \boxplus_1 T_1^{(t)}) \oplus \beta \cdot R_2, \\
\gamma \cdot z^{(t+1)} &= \gamma \cdot (\sigma(R_2 \boxplus_1 (R_3 \oplus T_2^{(t)})) \boxplus_1 T_1^{(t+1)}) \oplus \gamma \cdot E(R_1).
\end{align*}
\]
(2)

If we XOR the three masked expressions in Equation (2), then we get
\[
\begin{align*}
\alpha \cdot z^{(t-1)} &\oplus \beta \cdot z^{(t)} \oplus \gamma \cdot z^{(t+1)} = \beta \cdot (R_1 \boxplus_1 T_1^{(t)}) \oplus \gamma \cdot E(R_1) \\
&\quad \oplus \alpha \cdot (E^{-1}(R_2) \boxplus_1 T_1^{(t-1)}) \oplus \beta \cdot R_2 \oplus \alpha \cdot E^{-1}(R_3) \\
&\quad \oplus \gamma \cdot (\sigma(R_2 \boxplus_1 (R_3 \oplus T_2^{(t)})) \boxplus_1 T_1^{(t+1)}).
\end{align*}
\]
(3)

We approximate the non-linear operations in Equation (3) as
\[
\begin{align*}
\beta \cdot (R_1 \boxplus_1 T_1^{(t)}) &\oplus \gamma \cdot E(R_1) \boxplus m \cdot T_1^{(t)}, \\
\alpha \cdot (E^{-1}(R_2) \boxplus_1 T_1^{(t-1)}) &\oplus \alpha \cdot E^{-1}(R_3) \oplus \gamma \cdot (\sigma(R_2 \boxplus_1 (R_3 \oplus T_2^{(t)})) \boxplus_1 T_1^{(t+1)}) \\
&\equiv n \cdot T_1^{(t+1)} \oplus \beta \cdot R_2 \oplus h \cdot T_2^{(t)} \oplus l \cdot T_1^{(t-1)},
\end{align*}
\]
where \( l, m, n, \) and \( h \) are the 128-bit linear masks applied to \( T_1^{(t-1)}, T_1^{(t)}, T_1^{(t+1)}, \) and \( T_2^{(t)} \), respectively. We then get the noise of the overall linear approximation as below,
\[
N = \beta \cdot (R_1 \boxplus_1 T_1^{(t)}) \oplus \gamma \cdot E(R_1) \boxplus m \cdot T_1^{(t)} \oplus \alpha \cdot (E^{-1}(R_2) \boxplus_1 T_1^{(t-1)}) \oplus l \cdot T_1^{(t-1)} \oplus \beta \cdot R_2 \\
\quad \oplus \alpha \cdot E^{-1}(R_3) \oplus \gamma \cdot (\sigma(R_2 \boxplus_1 (R_3 \oplus T_2^{(t)})) \boxplus_1 T_1^{(t+1)}) \oplus n \cdot T_1^{(t+1)} \oplus h \cdot T_2^{(t)} \quad (4)
\]
\( c(N) = \sum_{\xi} c(N_1)c(N_2)c(N_3) \).

We introduce an intermediate linear mask \( \xi \) to \( R_2 \) and further divide the noise \( N \) into three parts, \( N_1, N_2, \) and \( N_3 \), defined as below as done in [ZFZ22]:
\[
\begin{align*}
N_1 &= \beta \cdot (R_1 \boxplus_1 T_1^{(t)}) \oplus \gamma \cdot E(R_1) \oplus m \cdot T_1^{(t)}, \\
N_2 &= \alpha \cdot (E^{-1}(R_2) \boxplus_1 T_1^{(t-1)}) \oplus \xi \cdot R_3 \oplus l \cdot T_1^{(t-1)} \oplus \alpha \cdot E^{-1}(R_3) \oplus h \cdot R_3, \\
N_3 &= \gamma \cdot (\sigma(R_2 \boxplus_1 (R_3 \oplus T_2^{(t)})) \boxplus_1 T_1^{(t+1)}) \oplus n \cdot T_1^{(t+1)} \oplus (\beta \oplus \xi) \cdot R_2 \oplus h \cdot (R_3 \oplus T_2^{(t)}).
\end{align*}
\]
(5)

As \( N_1 \) only involves variables \( R_1 \) and \( T_1^{(t)} \) that do not appear in \( N_2 \) and \( N_3 \), thereby establishing the independence of \( N_1 \) from \( N_2 \) and \( N_3 \). Noises \( N_2 \) and \( N_3 \) are not independent as they both involve variables \( R_2 \) and \( R_3 \). However, as \( R_3 \) in \( N_3 \) is XOR-ed with a random variable \( T_2^{(t)} \), it can be regarded as a new random variable. Therefore, we only need to exhaust the intermediate mask \( \xi \) to compute the correlation of the linear approximation noise by \( c(N) = \sum_{\xi} c(N_1)c(N_2)c(N_3) \) according to the correlation theorem.

It has been observed in [ZFZ22] that, for SNOW-V, one of the intermediate masks has a linear trail that dominates the linear hull, and therefore, \( c_{FSM} \approx c(N_1)c(N_2)c(N_3) \) for this specific \( \xi \). This conclusion was drawn from experimental results derived in [ZFZ22]. We have similar observations for the linear approximation derived for SNOW 5G, and we experimentally verified the dominance of the specific linear trail.
3.2 The Patterns of High-quality Linear Masks

We now analyze the general patterns of linear masks of high quality in SNOW 5G. We further split $N_2$ in Equation (5) into two parts, denoted as $P_1$ and $P_2$, as below,

\[ P_1 = \alpha \cdot E^{-1}(R_3) \oplus h \cdot R_3, \]
\[ P_2 = \alpha \cdot (E^{-1}(R_2) \boxplus_{16} T_1^{(t-1)}) \oplus \xi \cdot R_2 \oplus l \cdot T_1^{(t-1)}. \]

One can see that $P_1$ and $P_2$ are independent as they involve different variables; therefore, we have $c(N_2) = c(P_1)c(P_2)$. We can rewrite $P_1$ and $P_2$ by replacing $E^{-1}(R_3)$ and $E^{-1}(R_2)$ with $R_2^{(t-1)}$ and $R_1^{(t-1)}$, respectively, and then we get

\[ P_1 = \alpha \cdot R_2^{(t-1)} \oplus h \cdot E(R_2^{(t-1)}), \]
\[ P_2 = \alpha \cdot (R_1^{(t-1)} \boxplus_{16} T_1^{(t-1)}) \oplus \xi \cdot E(R_1^{(t-1)}) \oplus l \cdot T_1^{(t-1)}. \]

One can see that $N_1$ and $P_2$ have similar forms of linear approximations. We introduce 128-bit intermediate masks $\zeta_1$ and $\zeta_2$ for $N_1$ and $P_2$, respectively, and get

\begin{align*}
M_{1,\zeta_1} &= \beta \cdot (R_1 \boxplus_{16} T_1^{(t)}) \oplus \zeta_1 \cdot R_1 \oplus m \cdot T_1^{(t)}, \\
M_{2,\zeta_2} &= \zeta_1 \cdot R_1 \oplus \gamma \cdot E(R_1), \\
M_{3,\zeta_2} &= \alpha \cdot (R_1^{(t-1)} \boxplus_{16} T_1^{(t-1)}) \oplus \zeta_2 \cdot R_1^{(t-1)} \oplus l \cdot T_1^{(t-1)}, \\
M_{4,\zeta_2} &= \zeta_2 \cdot R_1^{(t-1)} \oplus \xi \cdot E(R_1^{(t-1)}). 
\end{align*}

We have $c(N_1) = \sum_{\zeta_1} c(M_{1,\zeta_1})c(M_{2,\zeta_2})$ and $c(P_2) = \sum_{\zeta_2} c(M_{3,\zeta_2})c(M_{4,\zeta_2})$. We further divide $N_3$ as done in [ZFZ22] as below

\begin{align*}
P_3 &= \gamma \cdot (R_2 \boxplus_{16} (R_3 \boxplus T_2^{(t)})) \boxplus_{16} T_1^{(t+1)} \oplus n \cdot T_1^{(t+1)} \oplus \Theta \cdot \sigma (R_2 \boxplus_{16} (R_3 \boxplus T_2^{(t)})), \\
P_4 &= \sigma(\Theta) \cdot (R_2 \boxplus_{16} (R_3 \boxplus T_2^{(t)})) \oplus (\beta \oplus \xi) \cdot R_2 \oplus h \cdot (R_3 \boxplus T_2^{(t)}),
\end{align*}

where $\Theta$ is a 128-bit intermediate mask, and then $c(N_3) = \sum_{\Theta} c(P_3)c(P_4)$.

3.2.1 Three Types of Linear Approximations

We now classify the linear approximations into three types and investigate the constraints and formats of the linear masks. Variables $X$ and $Y$ are 128-bit, and $U, V,$ and $W$ denote 128-bit linear masks.

**Type-I** $U \cdot X \oplus V_I \cdot E(X)$

**Type-II** $U_{II} \cdot (X \boxplus_{16} Y) \oplus V_{II} \cdot X \oplus W_{II} \cdot Y$

**Type-III** $U_{III} \cdot (X \boxplus_{16} Y) \oplus V_{III} \cdot E(X) \oplus W_{III} \cdot Y$

**Type-I Linear Approximation**

We now investigate how the AES round function affects the formats of related linear masks in Type-I linear approximations. We denote $X_i$ as the $i$-th byte of $X$ and analyze the linear masks corresponding to the AES encryption round function in detail. The AES encryption round function can be expressed as $E(X) = MixColumn \circ ShiftRow \circ S(X)$, where $S$ denotes the substitution layer consisting of 16 parallel 8-bit S-boxes. The step for adding a round key is not considered as it does not affect the linear approximation. Therefore, a Type-I linear approximation is indeed the XOR-sum of 16 independent linear approximations of S-boxes, and if one byte of $X$ is activated by $U_I$, the S-box of this byte should be activated by $V_I$ as well, vice versa.
If we use the matrix representation to denote $X$, we get the following expression:

$$E(X) = \text{MixColumn} \circ \text{ShiftRow} \circ S(X)$$

$$= \text{MixColumn} \circ \text{ShiftRow} \begin{bmatrix} S(X_0) & S(X_4) & S(X_8) & S(X_{12}) \\ S(X_1) & S(X_5) & S(X_9) & S(X_{13}) \\ S(X_2) & S(X_6) & S(X_{10}) & S(X_{14}) \\ S(X_3) & S(X_7) & S(X_{11}) & S(X_{15}) \end{bmatrix}$$

$$= \text{MixColumn} \begin{bmatrix} S(X_0) & S(X_4) & S(X_8) & S(X_{12}) \\ S(X_5) & S(X_9) & S(X_{13}) & S(X_1) \\ S(X_{10}) & S(X_{14}) & S(X_2) & S(X_6) \\ S(X_{15}) & S(X_3) & S(X_7) & S(X_{11}) \end{bmatrix}$$

$$= \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S(X_0) & S(X_4) & S(X_8) & S(X_{12}) \\ S(X_5) & S(X_9) & S(X_{13}) & S(X_1) \\ S(X_{10}) & S(X_{14}) & S(X_2) & S(X_6) \\ S(X_{15}) & S(X_3) & S(X_7) & S(X_{11}) \end{bmatrix}.$$  

If $U_i$ only introduces one active byte $X_i$ of $X$, then the column of $V_I$ that involves this byte of $\text{ShiftRow} \circ S(X)$ should be activated as well. For example, if $X_3$ is active, column 1 of $V_I$ should be active as well to activate $S(X_3)$. We represent the corresponding active columns of $V_I$ and $\text{ShiftRow} \circ S(X)$ as $(t_0, t_1, t_2, t_3)^T$ and $(w_0, w_1, w_2, w_3)^T$, respectively, and consider the expression $y$ as below:

$$y = (t_0, t_1, t_2, t_3) \cdot \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}^T.$$ \hspace{1cm} (6)

We would like the linear masks $(t_0, t_1, t_2, t_3)$ to introduce as few active S-boxes as possible, which means that $y$ involves the fewest variables of $S(X_j)$. When we target to have only one active S-box for Equation (6), it is equivalent to solving a system of linear equations to cancel out three variables and keep only one variable, e.g., $w_3$, as

\begin{align*}
& t_0 \cdot 02w_0 + t_1 \cdot 01w_0 + t_2 \cdot 01w_0 + t_3 \cdot 03w_0 = 0, \\
& t_0 \cdot 03w_1 + t_1 \cdot 02w_1 + t_2 \cdot 01w_1 + t_3 \cdot 01w_1 = 0, \\
& t_0 \cdot 01w_2 + t_1 \cdot 03w_2 + t_2 \cdot 02w_2 + t_3 \cdot 01w_2 = 0, \\
& t_0 \cdot 01w_3 + t_1 \cdot 01w_3 + t_2 \cdot 03w_3 + t_3 \cdot 02w_3 = \text{random.} 
\end{align*} \hspace{1cm} (7)

Such an equation system has four byte variables (i.e., linear masks) $t_0, t_1, t_2$, and $t_3$, to nullify $w_0, w_1, w_2$ with three exact equation constraints. Therefore, Equation (7) yields 255 possible solutions for the value of one column of $V_I$ if only one S-box is active per column. These 255 values are precisely the same as those derived in [GZ21], and so we fill the gap for the theoretical analysis of the observation.

We also note that when one column (say, column $j$) of $V_I$ activates only one S-box, i.e., this column of $V_I$ takes one of the 255 possible solutions, the active byte of $U_i$ can be located in any column. For example, if $j = 1$, the active byte of $X$ can be one of $X_4, X_9, X_{14}$, and $X_3$, which are located in different columns of $X$.

As the linear branch number of the $\text{MixColumn}$ operation in the AES round function is five, we can get the following proposition.

**Proposition 1.** For a Type-I linear approximation, if $i(1 \leq i \leq 4)$ bytes of $V_I$ in one column are equal to zero, which means that $4 - i$ bytes are non-zero, then $U_i$ must introduce at least $i + 1$ active bytes.
Type-II Linear Approximation

For a Type-II linear approximation $U_{II} \cdot (X \oplus_{16} Y) \oplus V_{II} \cdot X \oplus W_{II} \cdot Y$, the correlation can be computed as $c(U_{II}; V_{II}, W_{II}) = LA_{z_{15}} \cdots A_{z_{i}} A_{z_{0}} C$ where $L = (1, 0)$, $C = (1, 1)^T$ and $A_{z_{i}}(1 \leq i \leq 15)$ denotes the pre-computed connection matrix determined by $z_{i}$ which is computed by $z_{i} = U_{II} \cdot 2^{2} + V_{II} \cdot 2 + W_{II}$ [MJ05, NW06]. We present a brief introduction to how the connection matrices and correlation are computed in Appendix A, and for more details, we refer to [MJ05, NW06]. Furthermore, the correlation can be computed through an automaton illustrated in Figure 2 as proposed in [NW06]. The automaton has two states $e_{0}$ and $e_{1}$ representing $(1, 0)$ and $(0, 1)$, which are the possible intermediate states when computing $c(U_{II}; V_{II}, W_{II}) = LA_{z_{15}} \cdots A_{z_{1}} A_{z_{0}} C$ and multiplying the matrices from the left to right. The numbers in Figure 2 are decimal digits computed as the concatenation of the three bits of $U_{II}, V_{II}, W_{II}$ located at the same bit position, which is exactly $z_{i}$. The automaton starts from the highest bit to the lowest, and the absolute correlation is computed as $2^{-k}$, where $k$ is the number of transitions marked by a solid arrow in Figure 2. If the automaton enters state “0”, the correlation would stay zero. One can see that to have a non-zero correlation, the automaton must enter $e_{1}$ state through a number “7”, and this indicates the well-known constraint of linear masks in a linear approximation of a modular addition in Constraint 1. Appendix A briefly shows how the automaton is constructed and we also refer to [NW06] for more details.

Constraint 1. In a Type-II linear approximation, it is essential that the highest active bits in every $\oplus_{16}$-block of $U_{II}, V_{II}$, and $W_{II}$ are identical for the correlation to be non-zero. It implies that the position of the highest active bytes in each $\oplus_{16}$-block of $U_{II}, V_{II}$, and $W_{II}$ should also be the same.

![Figure 2: The automaton for the modular addition with two inputs](image)

Besides, one can easily derive the absolute correlation by counting the number of solid arrows in the automaton of a specific linear approximation without consuming much computation overhead. Generally, the more active $\oplus_{16}$-blocks are, the lower the correlation could be. Such a statement is not strictly correct. However, the number of $\oplus_{16}$-blocks, like the number of active S-boxes, can be used to get some information about the correlation heuristically. Besides, when the high bytes of the linear masks of a $\oplus_{16}$-block are fixed, the correlation will become smaller when any lower byte is also active as the solid lines in Figure 2 will become more. We use the maximum number of active bytes in each linear mask as one indicator for roughly evaluating the linear approximation of a 16-bit modular addition.

Type-III Linear Approximation

A Type-III linear approximation can be split into Type-I linear approximations and Type-II linear approximations by introducing an intermediate mask $M_{III}$, i.e.,
We have analyzed the possible numbers of active bytes and columns of $M_{III}$ and the possible numbers of introduced modular additions when $V_{III}$ has different active columns, as presented in Appendix B. We note that these numbers can also be helpful for Type-I linear approximations.

### 3.2.2 Constraints of Linear Masks to Have Potentially High Correlation

In this section, we analyze the constraints of linear masks in Equation (3) that can introduce high correlations. Some constraints are derived strictly following the analysis of the three basic linear approximations in Section 3.2.1, while some are obtained heuristically.

**Constraint 2.** The positions of the active $\boxplus_{16}$-blocks in $\gamma$, $n$, and $\Theta$ should be consistent, and the highest active byte positions should be the same.

**Constraint 3.** The positions of the active $\boxplus_{16}$-blocks in $\sigma(\Theta)$, $\beta \boxplus \xi$, and $h$ should be consistent, and the highest active byte positions should be the same.

**Constraint 4.** The positions of the active $\boxplus_{16}$-blocks in $\beta$, $\zeta_1$, and $m$ should be consistent, and the highest active byte positions should be the same.

**Constraint 5.** The positions of the active $\boxplus_{16}$-blocks in $\alpha$, $\zeta_2$, and $l$ should be consistent, and the highest active byte positions should be the same.

**Constraint 6.** The active byte positions of $\beta$ and $\xi$ are within the active $\boxplus_{16}$-blocks of $\sigma(\Theta)$.

Constraint 6 is not derived from a strict proof but from theoretical analysis and empirical knowledge for obtaining relatively high correlations. When $\beta$ or $\xi$ have active bytes of which the positions are outside the active $\boxplus_{16}$-blocks of $\sigma(\Theta)$, the other linear mask of $\beta$ and $\xi$ also needs active bytes at these positions to cancel out the activeness, otherwise, $c(P_1) = 0$ according to Constraint 3. However, when $\beta$ and $\xi$ have additional active bytes, $N_1$ will introduce more $\boxplus_{16}$-blocks and $N_2$ will introduce more $\boxplus_{16}$-blocks as well as S-boxes. Therefore, it is reasonable to restrict the positions of active bytes of $\beta$ and $\xi$ within the active $\boxplus_{16}$-blocks of $\sigma(\Theta)$.

Actually, we can further have a heuristic constraint that $h$ and $\xi$ have similar formats. This is because $h$ determines $\alpha$ in $P_1$ and $\xi$ determines $\zeta_2$ in $M_4(\Theta)$ both through Type-I linear approximations in the same way. Besides, $\alpha$ and $\zeta_2$ are involved in the same $\boxplus_{16}$-blocks in $M_4(\Theta)$; naturally, they have similar active byte positions. Therefore, $\xi$ and $h$ have similar formats. Our experiments also verified this.

With the abovementioned constraints, we propose indicators to evaluate and filter linear approximations. We define $\text{num}(x)$ as the number of active bytes in a 128-bit linear mask $x$, $\text{num}(i, x)$ as the number of active bytes in the $i$-th $\boxplus_{16}$-block of $x$, and $\text{nac}(x)$ as the number of active columns in $x$.

For a Type-I linear approximation $F = a \cdot X \oplus b \cdot E(X)$, we use $\text{num}_S(a, b)$ or $\text{num}_S(F)$ to represent the number of active S-boxes of the linear approximation and use it to evaluate the quality of $F$ roughly. The linear approximation is opted to be better when the value is smaller.

For a Type-II linear approximation $F = a \cdot (X \oplus_{16} Y) \oplus b \cdot X \oplus c \cdot Y$, we use $\text{num}_\oplus(F)$ or $\text{num}_\oplus(a, b, c)$ to denote the sum of the maximum numbers of active bytes in every $\oplus_{16}$-block as an indicator of the quality of $F$, i.e.,

$$\text{num}_\oplus(a, b, c) = \sum_{i=0}^{7} (\max\{\text{num}(i, a), \text{num}(i, b), \text{num}(i, c)\}).$$
The linear approximation can have a higher correlation when the value is smaller. This is because when a \( E_{16} \) operation has more active bytes, it is likely to have a lower correlation according to the automaton in Figure 2.

Now we re-categorize the linear approximation into two parts, say \( N_1 \oplus P_3 \) and \( N_2 \oplus P_4 \). Such re-categorization is because the correlations of \( N_1 \) and \( P_3 \) are greatly influenced by \( \gamma \), while \( N_2 \) and \( P_4 \) are influenced by \( \xi \) and \( h \). Then we get

\[
\text{num}_S(N_1 \oplus P_3) = \text{num}_S(\zeta_1, \gamma) = \text{num}(\zeta_1),
\]

\[
\text{num}_S(N_2 \oplus P_4) = \text{num}_S(\zeta_2, \xi) + \text{num}(\alpha, h) = \text{num}(\zeta_2) + \text{num}(\alpha).
\]

**Constraint 7.** The positions of active bytes of \( \beta \) and \( m \) are within the active byte positions of \( \zeta_1 \).

We make such constraint because \( \beta \) and \( m \) should have the same active \( E_{16} \)-blocks as \( \zeta_1 \) according to Constraint 1, and if \( \zeta_1 \) has only one active byte in an active \( E_{16} \)-block, \( \beta \) and \( m \) do not need to have two bytes as that will absolutely make the correlation lower according to Figure 2. Then we get \( \max\{\text{num}(i, \beta), \text{num}(i, \zeta_1), \text{num}(i, m)\} = \text{num}(i, \zeta_1) \) for \( 0 \leq i \leq 7 \).

We restrict \( \max\{\text{num}(i, \gamma), \text{num}(i, \Theta), \text{num}(i, n)\} = \text{num}(i, \gamma) \) for \( 0 \leq i \leq 7 \) in \( P_3 \), where \( \Theta \) is the intermediate mask, as it is likely to cancel out more S-boxes when \( \gamma \) has more active bytes in one column. Besides, \( \alpha \) and \( \zeta_2 \) have the same active byte positions as analyzed before. According to the linear branch number of the \( \text{MixColumn} \) operation in the AES round function, we have

\[
\text{num}_E(N_1 \oplus P_3) = \sum_{i=0}^{7} (\text{num}(i, \gamma) + \text{num}(i, \zeta_1)) = 5 \times \text{nac}(\gamma),
\]

\[
\text{num}_E(N_2 \oplus P_4) = \sum_{i=0}^{7} (\text{num}(i, \alpha) + \max\{\text{num}(i, \sigma(\Theta)), \text{num}(i, \beta \oplus \xi), \text{num}(i, h)\})
\]

\[
\geq \sum_{i=0}^{7} (\text{num}(i, \alpha) + \text{num}(i, h)) = 5 \times \text{nac}(h).
\]

Here we use the best case \( 5 \times \text{nac}(h) \) to represent \( \text{num}_E(N_2 \oplus P_4) \).

### 3.3 Divide-and-Conquer Method to Search for High-quality Linear Masks

With the constraints of linear masks analyzed in Section 3.2.2, we now search for high-quality masks through deep theoretical analysis and little computation overhead. We divide the linear approximations into several cases according to the number of active bytes of the critical intermediate linear mask \( \Theta \) and analyze the correlation in each case. The critical intermediate linear mask could originally be \( \xi \) as we need to compute \( c(N) \) by exhausting \( \xi \), but \( \xi \) has too many possibilities that are not easy to classify. Instead, we consider the different cases of \( \Theta \) due to two main reasons: (1) the number of active bytes of \( \Theta \) greatly influences the correlation so we only need to consider few different cases of \( \Theta \); (2) once the active bytes of \( \Theta \) are determined, the active modular additions in \( P_3 \), \( P_4 \) and formats of other linear masks, e.g., \( \gamma, n, h, \xi, \beta \), can be roughly determined.

1. **\( \Theta \) has one active byte.**

When \( \Theta \) has one active byte, the number of active bytes of \( \gamma \) is limited to one or two according to Constraint 2 and \( P_3 \). Correspondingly, \( \xi_1 \) can have three to four bytes. These bytes are located in different columns; thus, three to four \( E_{16} \)-blocks of \( \beta \) at different
columns are activated, which contradicts Constraint 6. Therefore, we do not dig into this case.

Figure 3: One example when Θ has one active byte.

As an illustration in Figure 3, assuming that Θ is active only at byte 1, σ(Θ) is active at byte 4. As a consequence, γ is only active at byte 1 or bytes 1 and 0 according to P3, while at least three of bytes 0, 5, 10, and 15 of ζ1 and β are active according to M1,ζ1 and M2,ζ1. It implies that at least two bytes outside the active ⊕16-block of σ(Θ) are active, which contradicts Constraint 6.

2. Θ has two active bytes.

This case can be further classified into three small cases.

Case 2.1 The active bytes of Θ are in the same row

When the two active bytes of Θ are in the same row, γ will activate two ⊕16-blocks located in different columns, and there are one or two active bytes in each column according to P3. As a result, ζ1 has 2 × 3 to 2 × 4 active bytes, and these bytes will be distributed in three or four different columns according to Table 4. This indicates that the active bytes of β are spread over at least three active columns based on M1,ζ1. However, the two active bytes of σ(Θ) are located in one active column, which contradicts Constraint 6. Therefore, this case will not happen.

Figure 4 presents an example of this case when Θ is active at bytes 1 and 5, then σ(Θ) will be active at bytes 4 and 5. γ has 2 ~ 4 active bytes according to P3 and ζ1 should have at least three active bytes in bytes 0, 5, 10, and 15 and three active bytes in bytes 3, 4, 9, and 14. As a consequence, at least two active bytes are outside the active ⊕16-block of σ(Θ), which contradicts Constraint 6.

Figure 4: An example when Θ has two active bytes located in the same row

Case 2.2 The two active bytes of Θ are in the same column

In this case, the active bytes of Θ can be located within one single ⊕16-block or two ⊕16-blocks in the same column, i.e., nac(γ) = 1. Meanwhile, the two active bytes of σ(Θ) are in two ⊕16-blocks in the same row. According to P3, γ has only one active column, then ζ1 and β have active bytes in different columns. Then we get that ζ1 and β have the same active byte positions, and each active column contains one active byte according to M1,ζ1. As σ(Θ) are active at two columns, β can have one or two active bytes according to Constraint 6, then the two active bytes of Θ are in different ⊕16-blocks. Otherwise, γ will only have two active bytes, and ζ1, β will have active bytes in at least three columns. Therefore, the number of active bytes in γ is three or four, and we get that \( \text{num}(N_1 \oplus P_3) = 5 \times nac(γ) = 5 \times 1 = 5 \), \( \text{num}(N_2 \oplus P_3) = 5 \times nac(h) = 5 \times 2 = 10 \). Then

\[
\text{num}(N_1 \oplus P_3) + \text{num}(N_2 \oplus P_3) = 15.
\]
As $\sigma(\Theta)$ has two active $\boxplus_{16}$-blocks located in two columns, $h$ and $\xi$ each will have $1 \sim 2$ active bytes at these two columns, then $\alpha$ and $\zeta_2$ will each have six active bytes in the best case. Therefore, $\text{num}_S(N_2 \oplus P_1) = 2 \times 6 = 12$.

When $\gamma$ has three active bytes, $M_{2,\xi_1}$ can introduce two S-boxes, in this case $\text{num}_S(N_1 \oplus P_3) + \text{num}_S(N_2 \oplus P_4) = 2 + 12 = 14$. Figure 5 presents the possible cases when the two active bytes of $\Theta$ are in the lower and higher positions of two $\boxplus_{16}$-blocks, respectively. We have $\text{num}_S(N_1 \oplus P_3) = 2$.

When $\gamma$ has four active bytes, the active bytes of $\Theta$ should be in the high positions of the two $\boxplus_{16}$-blocks, and $M_{2,\xi_1}$ can introduce only one S-box. In this case $\text{num}_S(N_1 \oplus P_3) + \text{num}_S(N_2 \oplus P_4) = 1 + 12 = 13$. The column 1 of $\beta \oplus \xi$ contains two active bytes according to Figure 6. We have $\text{num}_S(N_1 \oplus P_3) = 2$.

**Figure 5:** Different cases when the two active bytes of $\Theta$ are at the low and high byte positions of two $\boxplus_{16}$-blocks

**Figure 6:** Different cases when the two active bytes of $\Theta$ are at the higher byte positions of two $\boxplus_{16}$-blocks

**Case 2.3 The two active bytes of $\Theta$ are in the different columns and rows**

If the two active bytes of $\Theta$ are located in different rows and columns, the active bytes of $\gamma$ are distributed across two $\boxplus_{16}$-blocks, each with one or two active bytes. $\zeta_1$ and $\beta$ have $2 \times (3 \sim 4)$ active bytes located in least three columns according to Table 4. However, the active bytes of $\beta$ are distributed in at most two columns as $\sigma(\Theta)$ are located in two columns. Therefore, this case will not happen.

Figure 7 illustrates one example of this case. When bytes 4 and 13 of $\Theta$ are active, $\gamma$ contains two active $\boxplus_{16}$-blocks. Thus, the active bytes of $\beta$ are distributed in at least three columns. However, bytes 1 and 7 of $\sigma(\Theta)$ are active, and $\beta$ is only active in columns 0 and 1, which results in a contradiction.
3. Θ has three active bytes.

Case 3.1 The active bytes of Θ are in the same column
If the three active bytes of Θ are in the same column, three active bytes of σ(Θ) would be in the same row. Therefore, γ has one active column with 3 ∼ 4 active bytes and at least two Θ₁6-blocks are activated in P₃. We note that we always assume γ has more active bytes than Θ in one Θ₁6-block based on the analysis in Section 3.2.2. Besides, three Θ₁6-blocks will be activated in P₄, and h will have three active columns with one or two active bytes in each column. As a consequence, ζ₁ would have at least one active byte based on M₂ζ₁, and α and ζ₂ can each have 3 × 3 = 9 S-boxes.

We can then compute that numₑ(N₁ ⊕ P₃) + numₑ(N₂ ⊕ P₄) = 5 × 1 + 5 × 3 = 20, and numₛ(N₁ ⊕ P₃) + numₛ(N₂ ⊕ P₄) = 1 + 9 + 9 = 19. As the two numbers are both much higher than those in Case 2.2, respectively, we conjecture that the correlation would be lower than that derived there, and we do not dig deep into this case.

Case 3.2 The active bytes of Θ are located in at least two columns
If the three active bytes of Θ are located in at least two columns, γ would have at least one active Θ₁6-block in each column. Besides, at least one column has 1 ∼ 2 active bytes. Then β has at least three bytes spread over at least three columns according to N₁. As a result, the three active bytes of σ(Θ) are distributed across three columns. Otherwise, it contradicts Constraint 6. Then h and ξ both have three active columns each with 1 ∼ 2 active bytes, and introduce at least 3 × 3 × 2 = 18 S-boxes in total. In a better case, γ can have only two columns, one with 1 ∼ 2 active bytes and the other with 2 ∼ 4 active bytes. Then ζ₁ can introduce at least 3 + 1 = 4 active S-boxes. We have numₑ(N₁ ⊕ P₃) + numₑ(N₂ ⊕ P₄) = 5 × 2 + 5 × 3 = 25, numₛ(N₁ ⊕ P₃) + numₛ(N₂ ⊕ P₄) = 4 + 18 = 22. We refuse it because the numbers are much higher than those in Case 2.2.

4. Θ has four active bytes.

Case 4.1 The active bytes of Θ are in the same column
When the four active bytes of Θ are in the same column, γ has one active column with 2 ∼ 4 active bytes according to P₃. σ(Θ) has four active bytes located in four different columns. Thus, h has four active columns, and each column has 1 ∼ 2 active bytes according to P₄. Therefore, α and ζ₂ each has at least 3 × 4 active bytes, and ζ₁ has at least one active byte according to M₂ζ₁.

Then we can compute that numₑ(N₁ ⊕ P₃) + numₑ(N₂ ⊕ P₄) = 5 × 1 + 5 × 4 = 25, and numₛ(N₁ ⊕ P₃) + numₛ(N₂ ⊕ P₄) = 1 + num(α) + num(ζ₂) = 13 + 13 + 12 = 26. We do not dig into this case as these numbers are much higher than those in Case 2.2, and the correlation is expected to be lower.

Case 4.2 The active bytes of Θ are located in two columns
If the four active bytes of Θ are located in two columns, γ would have two active columns. The optimal case is that each column of γ has four active bytes and introduces only one S-box in M₂ζ₁; therefore, the theoretical minimum value of num(ζ₁) is two, as illustrated in Figure 8. Linear mask h can also have only two active columns; each column
has 2 ~ 4 active bytes. Therefore, the minimums of $num(\zeta_1)$, $num(\alpha)$, and $num(\zeta_2)$ are all two.

![Figure 8: An example when the four active bytes of $\Theta$ are located in two columns](image)

We have $num_{\oplus}(N_1 \oplus P_3) + num_{\oplus}(N_2 \oplus P_4) = 5 \times 2 + 5 \times 2 = 20$, and $num_{S}(N_1 \oplus P_3) + num_{S}(N_2 \oplus P_4) = 2 + num(\alpha) + num(\zeta_2) = 2 + 2 = 6$. The value $num_{\oplus}(N_1 \oplus P_3) + num_{\oplus}(N_2 \oplus P_4)$ is larger than that in Case 2.2, however, $num_{S}(N_1 \oplus P_3) + num_{S}(N_2 \oplus P_4)$ is smaller than that in Case 2.2, so we also take it into account for further exploration.

**Case 4.3 The active bytes of $\Theta$ are located in three or more active columns**

When the active bytes of $\Theta$ are located in three or more active columns, $\gamma$ will have at least three active columns, and at least one column has only one $\oplus_{16}$-block. As a result, the active bytes of $\beta$ will be spread over at least three columns. Therefore, $\sigma(\Theta)$ and $h$ will have three active columns. Then $num_{\oplus}(N_1 \oplus P_3) + num_{\oplus}(N_2 \oplus P_4) \geq 5 \times 3 + 5 \times 3 = 30$. As $\gamma$, $h$, and $\xi$ all have three active columns, even if each column only introduces one active S-box, the total number of S-boxes would be 9. The numbers are already higher than those in Case 4.2; therefore, we do not dig into this case.

5. $\Theta$ has five or more active bytes.

When $\Theta$ has five or more active bytes, these bytes will be located in two or more columns, and the sum of $nac(\gamma)$ and $nac(h)$ cannot be less than five. If we calculate $num_{\oplus}(N_1 \oplus P_3)$ and $num_{\oplus}(N_2 \oplus P_4)$, their sum would be not less than $5 \times 2 + 5 \times 3 = 25$. The minimum value of $num(\zeta_1)$ is two as $\gamma$ has at least two active columns, while $num(\alpha)$ and $num(\zeta_2)$ are both larger than two as $h$ and $\xi$ also have at least two active columns. Therefore, $num_{S}(N_1 \oplus P_3) + num_{S}(N_2 \oplus P_4) > 6$. We do not dig into this case because these numbers are higher than those in Case 4.2.

In summary, we can mainly focus on digging into Case 2.2 and Case 4.2, and other cases can be sifted out as they are likely to have lower correlations.

### 3.4 Experimental Results

#### 3.4.1 Case 2.2

We take the first case in Figure 6 as an example to describe the process for searching high-quality linear masks for Case 2.2. We first note that the correlations of linear approximations of different 16-bit modular additions or S-boxes can be computed in parallel.

**Step 1. Determine the initial range of active columns of $\gamma$ according to the analysis in Section 3.3.**

In Figure 6, when $\Theta$ has two active bytes located in column 1, $\gamma$ has one active column and 2 ~ 4 active bytes among bytes 4, 5, 6, 7. Therefore, $\gamma$ can take one of the 255 candidates according to Equation (7) to only activate one S-box in $M_{2,\zeta_1}$, i.e., $\zeta_1$ and $\beta$ have only one active byte.

**Step 2. Choose $\gamma$ and $n$ according to $|e(P_3)|$**

For every 255 possible choices of $\gamma$ determined in Step 1, we introduce Algorithm 1 to choose good candidates of $\gamma$ and $n$ according to $|e(P_3)|$. For example, when we choose
\( \gamma = 0x81ec5a80 \), the two active bytes of \( \Theta \) are limited to the 5-th and 7-th positions; otherwise, \( c(P_3) \) becomes zero. Algorithm 1 computes the correlation of a \( \Xi_{10} \)-block based on Figure 2 under given input and output linear masks. We note that the high \( \Xi_{10} \)-block and the low \( \Xi_{10} \)-block in one column can be computed in parallel. We store the pairs of \( \gamma \) and \( n \) that introduce the 1490 highest values of \( c(P_3) \). The complexity of this step is \( 255 \times 2^{16} \times 2^8 \approx 2^{32} \).

**Step 3. Select **\( \gamma, n, h \) **and \( \beta \oplus \xi \) according to** \( |c(N_3)| \)

As \( \sigma(\Theta) \) has two active bytes located in two columns, \( h \) contains two active bytes in one column, and each column introduces three active bytes in \( \alpha \). Every such three active bytes of \( \alpha \) can have four possibilities of locations, and each possibility can have 255 values (three byte positions are fixed, and the remaining position can have any non-zero value). We obtain the initial candidates of \( \gamma \) and \( n \), \( \alpha \) and \( \xi \) have 24 and 2 possibilities, respectively, in the stored table derived in **Step 3**, we select good candidates of \( |c(N_3)| \). We note that here we make the constraint that the active byte of \( \beta \) has the same active positions. For every combination of \( h, \xi, \gamma, n \), we exhaust \( \Theta \) to compute \( c(N_3) = \sum_{\Theta} c(P_3)c(P_4) \). We then select and store the candidates \( \gamma, n, h, \beta \) and \( \xi \) that give decent \( |c(N_3)| \). We note that here we expect the constraint that the active byte of \( \beta \) has the same value as the corresponding byte of \( \xi \) to restrict the searching space. This is also conform to the heuristic observation that the correlation would be higher when the active bytes are fewer in \( P_4 \). The complexity of this step is \( 4 \times 255^2 \times 1490 \times 2^8 \approx 2^{36.5} \).

**Step 4. Select candidates of** \( m \) **based on** \( |c(N_1)| \)

Given \( \beta \) and \( \gamma \) values selected in **Step 3**, we use Algorithm 1 to select \( m \) that gives decent \( |c(N_1)| \).

**Step 5. Select candidates of** \( \alpha \) **and** \( l \) **based on** \( |c(N_2)| \)

For each pair of \( h \) and \( \xi \) (columns 1 and 3 of \( h \) have 1 and 2 possibilities, and columns 1 and 3 of \( \xi \) have 24 and 2 possibilities, respectively, in the stored table derived in **Step 3**), we select good candidates of \( \alpha \) and \( l \) according to \( |c(N_2)| \). Given the values of \( h \), we exhaust the active byte of \( \alpha \) and compute \( c(P_1) \). After that we exhaust the same active byte of \( l \) and \( \xi \) to compute \( c(N_2) \) using \( c(N_2) = c(P_1)c(P_2) = c(P_1)\sum_{\xi_2} c(M_{3,\xi})c(M_{4,\xi}) \). We select high-quality \( \alpha \) and \( l \) based on \( |c(N_2)| \). The active 16-bit modular additions in \( M_{3,\xi} \) typically only have one active byte each, and \( P_1 \) and \( M_{4,\xi} \) can be divided into byte linear approximations as well. Therefore, we can search the linear masks in bytes in parallel, and the complexity is around \( 3 \times 26 \times 2^4 \approx 2^{30.3} \).

**Step 6. Determine** \( \gamma, n, h, \xi, \beta, m, \alpha, \) **and** \( l \) **based on** \( |c(N)| \)

Compute \( c(N) = c(N_1)c(N_2)c(N_3) \) and choose decent \( \gamma, n, h, \xi, \beta, m, \alpha, \) and \( l \).

Through the steps above, the optimal result derived for this case is as below:

\[
\begin{align*}
\gamma &= 0, 0, 0x81ec5a80, 0 \\
n &= 0, 0, 0x81ec5a00, 0 \\
h &= 0xe1a0, 0, 0x7f40, 0 \\
\xi &= 0x8180, 0, 0x7f40, 0 \\
\beta &= 0, 0, 0x40, 0 \\
m &= 0, 0x40, 0 \\
\alpha &= 0x2, 0x000600, 0x20041, 0xb0000000 \\
l &= 0x22, 0x80006000, 0x20061, 0xf0000000 \\
c(N_1) &= -2^{-4.11126}, c(N_2) = -2^{-42.0545}, c(N_3) = -2^{-21.50815}, \\
\text{and} \\
c(N) &= -2^{-67.67}.
\end{align*}
\]

**Comments about the dominant trail.** We find that the linear trail we derive has a dominant role in contributing to the correlation. Actually, the total correlation should
be computed by $c(N) = c(N_1) \sum_\Theta c(N_2) \sum_\Theta c(P_3)c(P_4)$, i.e., we have two internal linear masks, $\Theta$ and $\xi$. We can handle the two linear masks differently. Now we assume that the outer masks $\alpha$, $\beta$, $\gamma$, $n$, $h$, $l$, and $m$ are fixed, and check the correlations of the other linear trails under the same outer linear masks. If $\Theta$ only has active bytes 5 and 7, then we have $c(N_3) \neq 0$; otherwise, $c(N_3) = 0$. When computing the correlation of $c(N_3)$, we have actually already exhausted $\Theta$, so we only need to consider the other linear trails introduced by $\xi$. Since $h = 0xe1a0, 0, 0x7f40, 0$, the active bytes of $\alpha$ and $\zeta_2$ are bytes 3, 4, 6, 9, 11, and 12. Therefore, the range of all possible values of $\xi$ in one column would shrink to 255 values. We compute $|c(N_2)c(N_3)|$ for all the $255 \times 255$ $\xi$ values, and observe that there exists one trail of which the correlation is much higher than others, which is exactly the one we derived. The suboptimal linear trail is obtained when $\xi = 0xe1a0, 0, 0x7f40, 0$, and $c(N_2) = 2^{-43.17286}, c(N_3) = -2^{-27.09346}, c(N) = 2^{-74.38}$.

One can see that the suboptimal linear trail has a much lower correlation than the one we derived.

### 3.4.2 Case 4.2

When $\Theta$ has four active bytes, Figure 8 is likely to be the best case. In this case, the active byte positions of $\Theta$ and $\sigma(\Theta)$ are the same, and thus, $\gamma$ and $h$ can have two full columns (columns 1 and 3) to introduce as few S-boxes as possible. We now focus on one part of the linear masks as below:

\[
\begin{align*}
\gamma_47 \cdot \gamma_40 \cdot \gamma_39 \cdots \gamma_{32}, \\
n_{47} \cdot n_{40} \cdot n_{39} \cdots n_{32}, \\
\Theta_{47} \cdots \Theta_{40}, 00000000 \\
\sigma(\Theta)_{47} \cdots \sigma(\Theta)_{40}, 00000000, \\
h_{47} \cdots h_{40}, h_{39} \cdots h_{32}, \\
(\beta \oplus \xi)_{47} \cdots (\beta \oplus \xi)_{40}, (\beta \oplus \xi)_{39} \cdots (\beta \oplus \xi)_{32}.
\end{align*}
\]

Formats of bytes 4 of $\gamma$ and $h$, $n$ and $\beta \oplus \xi$ should be similar according to the automaton in Figure 2 since $\Theta_{47} \cdots \Theta_{40} = \sigma(\Theta)_{47} \cdots \sigma(\Theta)_{40}$. Byte 4 of $\gamma$ is equivalent to byte 4 of $h$ for optimizing $c(N_3)$. As a result, the first columns of $h$ and $\gamma$ are identical if they target to activate as few S-boxes as possible, as if one byte is fixed, only one of the 255 values derived from Equation (7) is satisfied.

Similarly, bytes 14 of $\gamma$ and $h$ act the same in the automaton, then the third columns of $\gamma$ and $n$ are identical. Therefore, we have $\gamma = h$, and furthermore, $\alpha$ and $\zeta_1$ have the similar active byte positions. Besides, $\alpha$ and $\zeta_2$ have the similar active byte positions in every active $\mathbb{F}_{16}$-block. Therefore, we conjecture that $\xi$ also has the similar format as $h$ and $\gamma$. Our experiments also verified this, and this case introduces a dominant correlation.

The procedure for searching linear masks is similar to those in Section 3.4.1, with the constraint $\gamma = h = \xi$. We derive the optimal result of this case as below.

\[
\begin{align*}
\gamma &= 0x2161a1e0, 0, 0x81ec5a80, 0 \\
n &= 0x2161c1c0, 0, 0x81e85a00, 0 \\
h &= 0x2161a1e0, 0, 0x81ec5a80, 0 \\
\xi &= 0x2161a1e0, 0, 0x81ec5a80, 0 \\
\beta &= 0x20, 0, 0x40, 0 \\
m &= 0x30, 0, 0x40, 0 \\
\alpha &= 0x1, 0, 0xc, 0 \\
l &= 0x1, 0, 0x8, 0
\end{align*}
\]
We have
\[ c(N_1) = -2^{-9.0238}, \quad c(N_2) = 2^{-13.7008}, \quad c(N_3) = -2^{-52.205963}, \]
and
\[ c(N) = 2^{-74.93}. \]

**Comparison.** The linear approximations in Case 2.2 and Case 4.2 actually represent two different tradeoffs between the numbers of active S-boxes and 16-bit modular additions, as listed in Table 3.

<table>
<thead>
<tr>
<th>Case 2.2</th>
<th>( N_1 )</th>
<th>( N_2 )</th>
<th>( P_3 )</th>
<th>( P_4 )</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 S-box, 1 ( \oplus_{16} )</td>
<td>12 S-boxes, 6 ( \oplus_{16} )</td>
<td>2 ( \oplus_{16} )</td>
<td>2 ( \oplus_{16} )</td>
<td>13 S-boxes, 11 ( \oplus_{16} )</td>
<td></td>
</tr>
<tr>
<td>2 S-box, 2 ( \oplus_{16} )</td>
<td>4 S-boxes, 2 ( \oplus_{16} )</td>
<td>4 ( \oplus_{16} )</td>
<td>4 ( \oplus_{16} )</td>
<td>6 S-boxes, 12 ( \oplus_{16} )</td>
<td></td>
</tr>
</tbody>
</table>

In Case 2.2, \( \Theta \) has only two active bytes located at the higher byte positions of the two \( \oplus_{16} \)-blocks in column 1, as illustrated in Figure 6. Therefore, \( P_3 \) and \( P_4 \) both have two full active \( \oplus_{16} \)-blocks. Linear mask \( \gamma \) is chosen to be one of the 255 values and full active in the specific column to only active one S-box in \( N_1 \). \( \xi \) and \( h \) both have two active bytes in columns 1 and 3, and each introduces six S-boxes in \( P_1 \) and \( P_2 \), respectively. Therefore, this case is more likely to make the correlation of the \( \oplus_{16} \)-blocks higher.

While in Case 4.2, \( \Theta \) has four active bytes located at the higher byte positions of all the \( \oplus_{16} \)-blocks in columns 1 and 3, as illustrated in Figure 8. \( \gamma, \xi \) and \( h \) are full active in these two columns and taking values among the specific 255 choices to minimize the numbers of S-boxes. Therefore, \( N_1, P_1, \) and \( P_2 \) each has only two active S-boxes. In total, there are only six active S-boxes in this case. On the other hand, \( P_3 \) and \( P_4 \) both have four full active \( \oplus_{16} \)-blocks. Therefore, this case is more likely to minimize the number of active S-boxes instead.

**3.4.3 Comparison with the results in [MJS+23]**

The best linear trail found in [MJS+23] has a correlation of \( 2^{-75.82} \), and our results improve it by a factor of \( 2^{1.15} \). The main fact allowing for this improvement is attributed to the larger space of linear masks we have considered. The linear trail derived in [MJS+23] is from automatic searching, while the searching space is too large to be exhausted. Therefore, the results in [MJS+23] are derived from a restricted searching space. Our analysis mainly relies on theoretical analysis and traverses the different cases of a critical linear mask (and correspondingly, the overall linear masks), which is the main reason that introduces better trails.

**3.5 Correlation Attack against SNOW 5G**

We now use the linear approximation of correlation \( c(N) = -2^{-67.67} \) derived in Section 3.4 to launch a correlation attack against SNOW 5G. The linear approximation can be expressed as
\[
\alpha \cdot z^{(t-1)} \oplus \beta \cdot z^{(t)} \oplus \gamma \cdot z^{(t+1)} \oplus l \cdot T_1^{(t-1)} \oplus m \cdot T_1^{(t)} \oplus n \cdot T_2^{(t+1)} \oplus h \cdot T_2^{(t)} = 0.
\]
A correlation attack tries to recover the key (the initial state after the initialization in the SNOW 5G case) by exploring such a correlation between the keystream and the output of the LFSR states. It is usually modeled as a decoding problem over \( \mathbb{F}_2^2 \) with the observed and reconstructed keystream symbols as a noisy version of the LFSR symbols through a discrete memoryless channel with a non-uniform noise which is related to the approximation noise.

Denote the initial state after the initialization as \( u = (u_{511}, u_{510}, \ldots, u_0) \) and a guessed value of it as \( \tilde{u} = (\tilde{u}_{511}, \tilde{u}_{510}, \ldots, \tilde{u}_0) \). Let \( L \) be the number of keystream words generated...
under a pair of key and IV. Each state bit of the LFSRs at any clock can be expressed as a linear combination of the initial state $u$ according to the feedback polynomials of the LFSRs. Therefore, one can always find $\Gamma(t) \in \mathbb{F}_2^{512}$ such that $\Gamma(t) \cdot u = l \cdot T_1^{(t-1)} + m \cdot \Gamma(t) \oplus n \cdot T_1^{(t+1)} \oplus h \cdot T_2^{(t)}$ for $1 \leq t \leq L - 1$, where $l, m, n, h$ are the linear masks that we found. We can then construct a sample using the guessed initial state $\hat{u}$ at clock $t$ as

$$\Phi_t(\hat{u}) = \alpha \cdot z(t-1) \oplus \beta \cdot z(t) \oplus \gamma \cdot z(t+1) \oplus \Gamma(t) \cdot \hat{u}$$

$$= \alpha \cdot z(t-1) \oplus \beta \cdot z(t) \oplus \gamma \cdot z(t+1)$$

$$\oplus l \cdot T_1^{(t-1)} \oplus m \cdot T_1^{(t)} \oplus n \cdot T_1^{(t+1)} \oplus h \cdot T_2^{(t)} \oplus \Gamma(t) \cdot (u \oplus \hat{u})$$

We collect a sufficient number of such samples at different clocks and derive the empirical distribution of the sample sequence. The distribution would behave biased with correlation around $c(N) = -2^{-67.67}$ if the guessed value of the initial state $\hat{u}$ is correct, otherwise, it will be random-like.

The correlation attack is typically divided into two stages: a preprocessing stage to explore and generate as many parity check equations as possible; and an online processing stage during which the decoding is performed using the collected parity checks. We follow the process of the correlation attack in [SIZ+22] to recover the initial state after the initialization. As the procedure of the correlation attack is not our main contribution, we refer to [SIZ+22] for more details.

**Preprocessing Stage**

Solving the decoding problem over $\mathbb{F}_2^n$ involves high complexity when $n$ is large, and the classical solution to it is to transform the decoding problem to a much smaller finite field $\mathbb{F}_2^B$, where $B < n$. Let the most significant $B$ bits of the binary vectors $u$ and $\Gamma(t)$ be $u_h$ and $\Gamma_h(t)$, respectively. Similarly, the corresponding lower $512 - B$ bits are respectively denoted by $u_l$ and $\Gamma_l(t)$. For the expression $(\Gamma(t_1) \oplus \Gamma(t_2)) \cdot u$, $1 \leq t_1 < t_2 \leq L - 1$, we can express it as the XOR of the higher and lower parts

$$(\Gamma(t_1) \oplus \Gamma(t_2)) \cdot u = (\Gamma_h(t_1) \oplus \Gamma_h(t_2)) \cdot u_h + (\Gamma_l(t_1) \oplus \Gamma_l(t_2)) \cdot u_l. \tag{8}$$

During the preprocessing stage, we target to find 2-tuple clocks $t_1$ and $t_2$ such that $\Gamma_l(t_1) = \Gamma_l(t_2)$, and then $(\Gamma(t_1) \oplus \Gamma(t_2)) \cdot u = (\Gamma_h(t_1) \oplus \Gamma_h(t_2)) \cdot u_h$. In this case, the lower $512 - B$ bits of $u$ are canceled. We can compute and store each $\Gamma(t) \in \mathbb{F}_2^{512}$ for $1 \leq t \leq L - 1$, and the time and memory complexities of this process are both $O(L)$. After that, we apply the fast sorting algorithm to sort $\Gamma(t)$ satisfying $\Gamma_l(t_1) = \Gamma_l(t_2)$ with time complexity $O(L + L \log_2 L)$. We can then collect new samples involving only the highest $B$ bits of the initial state as below:

$$T(\hat{u}_h) = \alpha \cdot (z(t_1-1) \oplus z(t_2-1)) \oplus \beta \cdot (z(t_1) \oplus z(t_2)) \oplus \gamma \cdot (z(t_1+1) \oplus z(t_2+1))$$

$$\oplus (\Gamma_l(t_1) \oplus \Gamma_l(t_2)) \cdot \hat{u}_h.$$
We compute and store each $\Gamma^{(t)} \in \mathbb{F}_2^{512}$ for $1 \leq t \leq L - 1$, and the time and memory complexities of this process are both $O(L)$. After that, we apply the fast sorting algorithm to sort $\Gamma^{(t)}$ satisfying $\Gamma^{(t)}_1 = \Gamma^{(t)}_2$ with time complexity $O(L + \log_2 L)$. We can then build a number of parity checks by combining any two items having the same low 512 $B$ bits in the table. The total time complexity of the preprocessing stage is $O(L + \log_2 L)$, and the data and memory complexities are both $O(L)$.

**Processing Stage**

With enough parity checks collected, this stage targets to recover the most highest $B$ bits of the initial state. The basic idea is to check the balancedness of the empirical distribution of the constructed sample sequence $T(\hat{u}_h)$ for each guess $\hat{u}_h$.

Among the $M$ parity check equations, we denote the $j$-th one as $T(\hat{u}_h)_j$, for $1 \leq j \leq M$. For each guessed value $\hat{u}_h \in \mathbb{F}_2^{512}$, we construct the statistic

$$I(\hat{u}_h) = \sum_{j=1}^{M} (-1)^{T(\hat{u}_h)_j}$$

and predict $\hat{u}_h$ that maximizes $I(\hat{u}_h)$ as the correct one.

We can compute $I(\hat{u}_h)$ through Fast Walsh Transform (FWT) for each guessed value $\hat{u}_h \in \mathbb{F}_2^{512}$ as done in [SJZ+22] with time complexity $O(2^B + M)$. When $B > \log_2 M$, the processing phase can be further accelerated by classifying $M$ parity check equations and exhausting the least significant $(B - \lfloor \log_2 M \rfloor)$-bit initial state [SJZ+22]. In this case, the time complexity is $O(B^2)$. The memory complexity of this stage is $O(2^B)$.

The total time complexity of the attack is $O(L + \log_2 L + 2^B(\lfloor \log_2 M \rfloor + 1))$, the data complexity is $O(L)$, and the memory complexity is $O(\max(L, 2^B))$. The success probability of the decoding under different numbers of parity checks can be derived using the results in [Sel08].

Under the success probability $p_s = 0.999992$, we tried different choices of $B$ and derived that when $B = 262$, $L = 2^{265.4}$, $M = 2^{279.8}$, the total complexity is the lowest. In this case, the total time, memory, and data complexities are $2^{279.8}$, $2^{265.5}$, and $2^{265.4}$, respectively.

### 4 Conclusion and Future Work

This paper presents theoretical linear cryptanalysis of SNOW 5G, the candidate of international standard for confidentiality and integrity protection in 5G. Different from existing linear cryptanalysis techniques that typically rely on automatic searching tools or linear approximation tables, we perform the analysis from the theoretical point of view. Specifically, we divide the linear approximation into small components and investigate the connections and constraints between the linear masks. With the analysis, we derive the formats of high-quality linear masks and further employ a divide-and-conquer method to search for them. We find a linear approximation that gives a correlation of $-2^{-67.67}$ and further use it to launch a correlation attack against SNOW 5G with complexity $2^{279.8}$.

Our results are the best cryptanalysis results of SNOW 5G till now, and more importantly, the results provide more insights into the security of SNOW 5G through the theoretical analysis. Our next step is to apply the methods to other SNOW-family ciphers.

**Acknowledgments**

This work was supported by the National Natural Science Foundation of China under grant numbers 62202492 and 62372464.
References


[3rd] 3rd generation partnership project; technical specification group services and system aspects; security aspects for inter-access mobility between non 3GPP and 3GPP access network (release 8).


A Linear approximations of modular additions [NW06]

Denote the input linear masks of a modular addition with $k$ inputs as $w^{(1)}, w^{(2)}, \ldots, w^{(k)}$ and output linear mask as $u$. The correlation can be computed through the following process. Let $k > 1$ be a fixed integer, and $L$ and $C$ are two constant vectors of dimension $k$, where $L = (1, 1, \ldots, 1)$ and $C = (1, 0, 0, \ldots, 0)$. Let $A_0, A_1, \ldots, A_{2k+1-1}$ be $2^{k+1}$ matrices of dimension $k \times k$ computed as below

\[
(A_r)_{d,c} = 2^{-k} [x \in \mathbb{F}_2^k : u \cdot g(x,c) = x \cdot v, f(x,c) = d] - 2^{-k} [x \in \mathbb{F}_2^k : u \cdot g(x,c) \neq x \cdot v, f(x,c) = d],
\]

where

\[
r = u2^k + \sum_{h=1}^{k} v_h 2^{h-1}, \quad v = (v_1, \ldots, v_k), \quad x = (x_1, \ldots, x_k),
\]

\[
c, d \in \{0, \ldots, k - 1\},
\]

\[
f : \{0, \ldots, k - 1\}^2 \rightarrow \{0, \ldots, k - 1\}, \quad f(x,c) = \lfloor w_H(x) + c \rfloor / 2,
\]

\[
g : \{0, \ldots, k - 1\}^2 \rightarrow \{0, 1\}, \quad g(x,c) = (w_H(x) + c) \mod 2.
\]

The functions $f$ and $g$ are the carry and sum functions for the basic schoolbook method for adding $k$ integers in binary.
Theorem 1. [NW06] Let \( n \geq 1 \) be an integer and let \( (u; w^{(1)}, \ldots, w^{(k)}) \) be a linear approximation of addition modulo \( 2^n \) with \( k \) inputs. Let \( z = z_{n-1} \cdots z_1 z_0 \) be the word associated with the approximation. We then have

\[
\text{cor}(u; w^{(1)}, \ldots, w^{(k)}) = LA_{z_{n-1}} \cdots A_{z_1} A_{z_0} C.
\]

When the modular addition has only two inputs (the SNOW 5G case), the correlation can be computed using a much simpler way using the automaton proposed in [NW06]. In this case, the correlation \( \text{cor}(u; w^{(1)}, w^{(2)}) = L' A'_{z_{15}} \cdots A'_{z_1} A'_{z_0} C' \), \( L' = (1, 1), \) \( C' = (1, 0)^T \).

Let \( A_i = H_2 A'_{z_i} H_2^{-1}, \) \( 0 \leq i \leq 15 \), where \( H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \) is the \( 2 \times 2 \) Hadamard matrix. Additionally, the matrices \( L \) and \( C \) are transformed as \( L = L' H_2^{-1} \) and \( C = H_2 C' \), respectively. The correlation of \( u, \) \( w^{(1)}, \) and \( w^{(2)} \) is obtained through \( \text{cor}(u; w^{(1)}, w^{(2)}) = L A'_{z_{15}} \cdots A'_{z_1} A'_{z_0} C \), where \( L = (1, 0) \) and \( C = (1, 1)^T \), and

\[
A_0 = \frac{1}{2} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_1 = A_2 = -A_4 = \frac{1}{2} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \\
A_7 = \frac{1}{2} \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix}, \quad -A_3 = A_5 = A_6 = \frac{1}{2} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.
\]

Let \( e_0 = (1, 0) \) and \( e_1 = (0, 1) \). We have

\[
\begin{align*}
e_0 A_0 &= e_0, e_0 A_7 = e_1, \\
e_0 A_i &= 0, i \neq 0, 7, \\
e_1 A_0 &= e_1 A_5 = e_1 A_6 = \frac{1}{2} e_1, \\
e_1 A_1 &= e_1 A_2 = e_1 A_7 = \frac{1}{2} e_0, \\
e_1 A_3 &= -\frac{1}{2} e_1, \\
e_1 A_4 &= -\frac{1}{2} e_0.
\end{align*}
\]

Therefore, the correlation \( L A'_{z_{15}} \cdots A'_{z_1} A'_{z_0} C \) computed by multiplying the matrices from left to right can be described by the automaton in Figure 2.

### B Different cases of active columns of \( V_{III} \)

When \( V_{III} \) has only one active column, \( M_{III} \) can have \( 1 \sim 4 \) active bytes that are located in different columns, and correspondingly, \( 1 \sim 4 \) 16-bit modular additions can be activated.

Table 4, Table 5, and Table 6 present the possible numbers of active bytes of \( M_{III} \) (#\( M_{III} \)), located columns of these active bytes (#columns), and involved \( \boxplus_{16} \) blocks (#\( \boxplus_{16} \)), when \( V_{III} \) have two, three, and four active columns, respectively.

**Table 4:** The possible numbers of active bytes and \( \boxplus_{16} \) blocks when \( V_{III} \) have two active columns

<table>
<thead>
<tr>
<th>#( M_{III} )</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#columns</td>
<td>1 \sim 2</td>
<td>2 \sim 3</td>
<td>3 \sim 4</td>
<td>3 \sim 4</td>
<td>3 \sim 4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>#( \boxplus_{16} )</td>
<td>1 \sim 2</td>
<td>2 \sim 3</td>
<td>2 \sim 4</td>
<td>3 \sim 5</td>
<td>4 \sim 6</td>
<td>5 \sim 6</td>
<td>6</td>
</tr>
</tbody>
</table>

**Table 5:** The possible numbers of active bytes and \( \boxplus_{16} \) blocks when \( V_{III} \) have three active columns

<table>
<thead>
<tr>
<th>#( M_{III} )</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>#columns</td>
<td>1 \sim 3</td>
<td>2 \sim 4</td>
<td>2 \sim 4</td>
<td>2 \sim 4</td>
<td>3 \sim 4</td>
<td>3 \sim 4</td>
<td>3 \sim 4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>#( \boxplus_{16} )</td>
<td>2 \sim 3</td>
<td>2 \sim 4</td>
<td>3 \sim 6</td>
<td>3 \sim 6</td>
<td>4 \sim 7</td>
<td>4 \sim 8</td>
<td>5 \sim 8</td>
<td>6 \sim 8</td>
<td>7 \sim 8</td>
<td>8</td>
</tr>
</tbody>
</table>
Table 6: The possible numbers of active bytes and ⊕_{16} blocks when V_{III} have four active columns

<table>
<thead>
<tr>
<th>#M_{III}</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9 ~ 10</th>
<th>11 ~ 12</th>
<th>13 ~ 14</th>
<th>15 ~ 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>#columns</td>
<td>1 ~ 4</td>
<td>2 ~ 4</td>
<td>2 ~ 4</td>
<td>2 ~ 4</td>
<td>2 ~ 4</td>
<td>3 ~ 4</td>
<td>3 ~ 4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>⊕_{16}</td>
<td>2 ~ 4</td>
<td>3 ~ 5</td>
<td>3 ~ 6</td>
<td>4 ~ 7</td>
<td>4 ~ 8</td>
<td>5 ~ 8</td>
<td>6 ~ 8</td>
<td>7 ~ 8</td>
<td>8</td>
</tr>
</tbody>
</table>

C Algorithm for computing the correlation of a linear approximation of the 16-bit modular addition

Algorithm 1 computes the correlation of a specific linear approximation of the 16-bit modular addition under given linear masks. We can consider it as an implementation of using the automation. The value of $e$ is 0 or 1, corresponding to states $e_0$ and $e_1$, respectively, in Figure 2. The variable flag corresponds to $z_i (0 \leq i \leq 15)$ in Figure 2, with a range of values from 0 to 7.

Algorithm 1 The algorithm for computing the correlation of a linear approximation of the 16-bit modular addition

1: Input: 16-bit linear masks $a'$, $b'$, and $c'$
2: Output: the correlation of $a' \cdot (X \oplus_{16} Y) \oplus b' \cdot X \oplus c' \cdot Y$
3: Initialize $k \leftarrow 0$, $e \leftarrow 0$, $sign \leftarrow 0$
4: for $i$ from 15 to 0 do
5:   compute $flag = a_i \times 2^2 + b_i \times 2 + c_i$
6:   if $e = 0$ then
7:     if $flag = 7$ then
8:       $e = 1$
9:     else if $flag = 0$ then
10:    $e = 0$
11:   else
12:     exit(0)
13: else
14:   if $flag = 3$ or $flag = 4$ then
15:     $sign = sign + 1$
16:   if $flag = 0$ or $flag = 3$ or $flag = 5$ or $flag = 6$ then
17:     $e = 1$
18:   $k = k + 1$
19: else
20:   $e = 0$
21: $k = k + 1$
22: return $(-1)^{sign \mod 2} \times 2^{-k}$