Security Analysis of Subterranean 2.0

Ling Song[1], Yi Tu[2], Danping Shi[3] and Lei Hu[3]

[1] Jinan University, Guangzhou, China

[2] Nanyang Technological University, Singapore

[3] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences, China

songling.qs@gmail.com,tuyi0002@e.ntu.edu.sg,
shidanping@iie.ac.cn,hulei@iie.ac.cn

Abstract. Subterranean 2.0 is a cipher suite that can be used for hashing, authenticated encryption, MAC computation, etc. It was designed by Daemen, Massolino, Mehrdad, and Rotella, and has been selected as a candidate in the second round of NIST's lightweight cryptography standardization process. Subterranean 2.0 is a duplex-based construction and utilizes an extremely simple one-round permutation in the duplex. It is the simplicity of the round function that makes it an attractive target of cryptanalysis.
In this paper, we examine the one-round permutation in various phases of Subterranean 2.0 and specify three related attack scenarios that deserve further investigation: keystream biases in the keyed squeezing phase, state collisions in the keyed absorbing phase, and one-round differential analysis in the nonce-misuse setting. First, to facilitate cryptanalysis, we propose two size-reduced toy versions of Subterranean 2.0: Subterranean-m and Subterranean-s. Then we exploit the resemblance between the non-linear layer in the round function of Subterranean 2.0 and SIMON's round function to construct our models for searching characteristics to be used in the keystream bias evaluation and state collision attack. Our results show that there exists no linear trail under the constraint of data limit imposed by the designers with a minimal number of output blocks. This partially confirms the designers' claim on the bias of keystream. Regarding state collisions in keyed modes, we find useful characteristics of two toy versions with which forgery attacks can be mounted successfully. However, due to the time-consuming search, the security of Subterranean 2.0 against the state collision attack in keyed modes still remains an open question. Finally, we observe that one-round differentials allow to recover state bits in the nonce-misuse setting. By proposing nested one-round differentials, we obtain a sufficient number of state bits, leading to a practical state recovery with only 20 repetitions of the nonce and 88 blocks of data. It is noted that our work does not threaten the security of Subterranean 2.0.

Keywords: Subterranean 2.0 · one-round permutation · keystream bias · state collision · state recovery

1 Introduction

The deployment of small computing devices such as RFID tags, microcontrollers, sensor nodes, and smart cards is becoming more and more common. Alongside this, the need for lightweight cryptography that aims to provide security solutions tailored for such resource-constrained devices is increasing. In 2013, the National Institute of Standards and Technology (NIST) initiated a public process to solicit, evaluate, and standardize lightweight authenticated encryption and hashing schemes that are suitable for use in constrained environments [Nat19]. In 2018, a call for submissions was launched and 57 submissions were received in 2019, among which 56 and 32 submissions were selected in the first and second rounds respectively. At the current stage, public evaluations of the candidates are strongly encouraged.

Subterranean 2.0 [DMMR20] is a cipher suite that can be used for hashing, authenticated encryption, MAC computation, and stream encryption, etc. It was designed by Daemen, Massolino, Mehrdad, and Rotella and has been selected by NIST as a candidate for the second round. Subterranean 2.0 shares features with its predecessor Subterranean [CDGP93] which can be seen as a precursor to the Sponge construction [BDPVA11]. The features of Subterranean 2.0 are summarized below.

Prime-sized state. Subterranean 2.0 operates on a state of 257 bits which is small but still supports both hashing and authenticated encryption. It offers a security strength of 128 bits in keyed modes and 112 bits in unkeyed mode. In authenticated encryption where a nonce is used, the nonce should not repeat.

Duplex-based construction The duplex [BDPV11] plays a core role in Subterranean 2.0. On top of it, three functions were built, namely, Subterranean-XOF, Subterranean-deck, and Subterranean-SAE, where the latter two are keyed functions. The duplex absorbs/squeezes 32-bit blocks in keyed modes and 8-bit blocks in unkeyed mode.

One-round permutation. In the duplex, a lightweight one-round permutation is used. The round function operates at bit level and has algebraic degree 2. It has a minimum of substructures and ultimate weak alignment which prevents large classes of attacks.

Blank rounds used. Between different phases, 8 blank rounds are used to prevent measurable characteristics between the controllable input and output.

Efficient hardware implementation. Subterranean 2.0 is designed for hardware and offers a good option for environments that require lightweight crypto in hardware with high throughput requirements. Besides, it is very suitable for protection against differential power analysis such as masking and threshold implementations.

Due to the extremely simple round function, Subterranean 2.0 is an attractive target for cryptanalysis. In the design specification [DMMR20], the designers mainly investigated the security of state collisions in unkeyed absorbing and differential/linear properties of a multiple-round permutation. As a complement, Liu *et al.* [LIM19] conducted cube-based cryptanalysis of Subterranean-SAE by exploiting the low algebraic degree of the round function. Liu *et al.* showed that when the number of blank rounds is reduced to 4, one can mount a state recovery attack while in the nonce-misuse setting the state recovery attack becomes practical using 2^{13} blocks of data.

With respect to the simple one-round permutation of Subterranean 2.0, there are interesting attacks in different phases. Below, we list three related attacks in keyed modes that deserve further investigation.

1. **Linear bias of output blocks in keyed squeezing phase.** By design, there is no linear bias over three output blocks of Subterranean 2.0. It is believed by the designers that over four or more output blocks there does not exist measurable bias. Any analytical results that approve or disapprove of this conjecture can help understand the security of Subterranean 2.0.

2. **State collisions in keyed absorbing phase.** In keyed modes, state collisions may lead to attacks like forgeries. However, security analysis against such attacks is missing from the specification document of Subterranean 2.0.

3. **One-round differential analysis of Subterranean-SAE in the message processing phase.** In the phase of processing the message, when a duplex call is invoked, an output block is squeezed and an input block absorbed before and after the one-round permutation, respectively. In the case where nonce repeats, one-round differentials can be observed over successive calls of duplex. The threat of such one-round differentials is not clear.

Our contribution. In this paper, we examine the security of Subterranean 2.0 in the above three attack scenarios regarding its one-round permutation. In order to investigate the bias of keystreams and the state collision attack, it requires to find useful linear and differential trails. When carrying out differential/linear analysis of Subterranean 2.0, we face two challenges. The first is that the permutation has only one round, so no round-reduced version is available for facilitating the differential/linear analysis. The other is the "dependency" issue that cannot be avoided either in differential analysis or linear analysis. The round function of Subterranean 2.0 uses bit-wise operations which allow weak alignment and its non-linear layer exploits logic AND of neighbouring bits. As neighbouring ANDs share an input bit, the ANDs are not independent and such dependency makes the differential/linear analysis difficult.

To tackle the first challenge, we propose two toy versions of Subterranean 2.0 with reduced state size. We choose two prime numbers 193 and 97, and adapt other parameters accordingly. We then have two toy versions: *Subterranean-m* and *Subterranean-s*.

To handle the second challenge, we observe that the non-linear layer of the round function of Subterranean 2.0 can be represented by a SIMON-like function. SIMON [BSS+13] is a family of lightweight block ciphers and has been extensively analysed since its publication, such as exact differential/linear analyses in [KLT15]. Starting with the tool in [KLT15], we build our models for searching optimal differential/linear trails of Subterranean 2.0 or tightening the bounds of differential probability or linear correlation.

Regarding biases of keystreams, our results show that there exists no linear trail under the constraint of data limit imposed by the designers for four output blocks. When we increase the number of output blocks by 1, there are no better linear trails in Subterranean-s, which gives some confidence that there might be no better linear trails over more output blocks as well for Subterranean 2.0. Thus, the designers' claim on the bias of keystreams is partially supported by our results.

As for state collisions, the search for differential trails of Subterranean 2.0 is still hard due to its weak alignment and relatively large state size (compared to block ciphers like SIMON). However, we find good differential trails for the two toy versions, with which forgery attacks can be mounted successfully. When we increase the number of input blocks by 1, better trails are found for both toy versions. It seems that the resistance of Subterranean 2.0 against state collision attacks might be weaker than its resistance against linear attacks.

Finally, we exploit the one-round differentials to recover the state in the nonce-misuse setting. If the nonce repeats, one-round differentials observed in the message processing phase of Subterranean-SAE will leak some bits of the state due to the algebraic degree 2 of the round function. Further, we find out that ordinary one-round differentials can recover 41 bits at most. To enlarge the number of state bits that can be recovered, we propose *nested one-round differentials* where a one-round differential is prepended to another in a delicate way. As a result, a sufficient number of state bits can be recovered, which leads to a full state recovery and further a key recovery. The attack is practical and takes only 20 repetitions of the nonce and 88 blocks of data, which is much lower than the data complexity of Liu *et al.*'s attack [LIM19]. Our analysis shows that Subterranean-like constructions with quadratic one-round permutation must be used carefully in practice since the security crashes without nonce uniqueness.

Organization. The rest of the paper is organized as follows. Basic notations, the design of Subterranean 2.0 and two toy versions are introduced in Section 2. Section 3 highlights several properties of Subterranean 2.0 and the relation to three attack scenarios: keystream biases, state collisions, and state recovery in the nonce-misuse setting. Linear attacks and state collisions in the keyed modes are investigated in Section 4. Section 5 presents a state recovery attack utilizing one-round differentials in the nonce-misuse setting. Finally, we conclude the paper in Section 6.

2 Notations and Specification of Subterranean 2.0

In this section, we start by giving our notations and then briefly introduce Subterranean 2.0, including its round function, the duplex object and two keyed members: Subterranean-deck and Subterranean-SAE. To facilitate cryptanalysis of Subterranean 2.0, we introduce two toy versions: Subterranean-m and Subterranean-s. For more details of Subterranean 2.0, we refer the interested reader to the official specification [DMMR20] .

2.1 Notations

b	The size of the state
d	The factor used in π of the round function
$\overline{M}$	The string M padded to 33 bits with 10*
ΔZ	The difference of Z
$S^{\alpha}(x)$	Cyclic right shift of vector x by α bits. If α is negative, it means cyclic left shift.
$\lvert \cdot \rvert$	The length in bits
$\vert\vert$	Concatenation of bit strings

2.2 Round Function

The round function R operates on a b-bit state and consists of four bit-oriented steps: $R = \pi \circ \theta \circ \iota \circ \chi$. Let s denote the state and s_i the i-bit of s. Then for all $0 \leq i < b$,

$$\begin{aligned}\chi &: \; s_i \leftarrow s_i + (s_{i+1} + 1) \cdot s_{i+2},\\ \iota &: \; s_0 \leftarrow s_0 + 1,\\ \theta &: \; s_i \leftarrow s_i + s_{i+3} + s_{i+8},\\ \pi &: \; s_i \leftarrow s_{d \times i}.\end{aligned}$$

Here the addition and multiplication of state bits are in $\mathbb{F}_2$ and expressions in the index are taken modulo b. In Subterranean 2.0, $b = 257, d = 12$.

2.3 Duplex Object and Two Keyed Functions

2.3.1 Duplex Object

The Subterranean 2.0 suite is built upon a duplex object which is displayed in Figure 1. The duplex uses a single-round permutation made from R and has two functions: the duplex call and the output extraction, the latter of which is optional. The dupelx call applies the round function R and absorbs a string M of at most 32 bits. Before adding the string to the internal state, the string is padded to 33 bits with 10*. The 33 bits are then injected into the state $s_{12^{4i}}, 0 \leq i < 33$. Namely, the injection rate is 33 bits. Before the duplex call, one may extract 32 bits from the state, each of which is the sum of two state bits:

$$Z_i = s_{12^{4i}} + s_{-12^{4i}},$$

for $0 \leq i < 32$. The details of indices used for injection and extraction are shown in Table 1.

When the input is an empty string, the combination of the round function and the injection is denoted as R_ϵ for convenience in the figures.

Table 1: Indices used for injection and extraction

i	12^{4i}	-12^{4i}	i	12^{4i}	-12^{4i}	i	12^{4i}	-12^{4i}	i	12^{4i}	-12^{4i}	i	12^{4i}
0	1	256	8	64	193	16	241	16	24	4	253	32	256
1	176	81	9	213	44	17	11	246	25	190	67		
2	136	121	10	223	34	18	137	120	26	30	227		
3	35	222	11	184	73	19	211	46	27	140	117		
4	249	8	12	2	255	20	128	129	28	225	32		
5	134	123	13	95	162	21	169	88	29	22	235		
6	197	60	14	15	242	22	189	68	30	17	240		
7	234	23	15	70	187	23	111	146	31	165	92		

2.3.2 Subterranean-deck and Subterranean-SAE

The Subterranean 2.0 suite contains three functions: Subterranean-XOR, Subterranean-deck and Subterranean-SAE. Subterranean-XOF is designed to be used for unkeyed hashing, while Subterranean-deck and Subterranean-SAE are keyed functions. In this paper, we focus on the latter two.

Subterranean-deck takes as input an arbitrary-length key and a sequence of arbitrary-length strings and returns a bit string of arbitrary length, as shown in Figure 2. Hence, it can be used as a stream cipher, a MAC function or for key derivation. Subterranean-SAE, depicted in Figure 3, is designed for authenticated encryption. Below, a detailed description of Subterranean-SAE is given. With the description of Subterranean-SAE in mind, it requires little extra effort to follow the working procedures of Subterranean-deck.

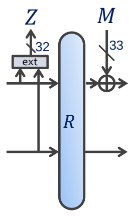

Figure 1: Duplex object

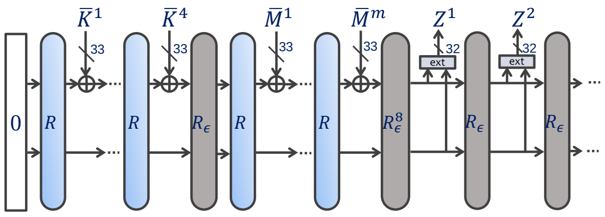

Figure 2: Subterranean-deck

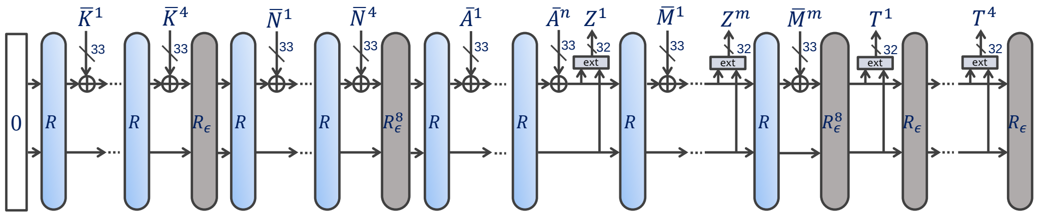

Figure 3: Subterranean-SAE

The input of Subterranean-SAE contains a 128-bit key, a 128-bit nonce N, an associated data (AD) A, and a message M and the output is composed of the ciphertext and a 128-bit tag T.

Processing the key At first, the state is initialized with 0. The 128-bit key is split into four 32-bit blocks K^1, K^2, K^3, K^4 and one empty block ϵ, as the last block should be strictly shorter than 32 bits. Each block is padded with 10* and the first four padded blocks are denoted by $\overline{K}^1$, $\overline{K}^2$, $\overline{K}^3$, and $\overline{K}^4$. The whole five blocks are then absorbed one by one through the duplex call.

Processing the nonce The nonce is split into 32-bit blocks with the last block being shorter than 32 bits. Pad each block with 10* and sequentially inject the padded blocks into the state in a series of duplex calls.

Processing the AD Invoke the duplex eight times, each with an empty message ϵ absorbed. Then absorb the AD in the same way as processing the nonce.

Processing the message The message is split into 32-bit blocks with the last block being shorter than 32 bits. Pad each block with 10*. Process message blocks one after another by the following steps: extract 32 output bits, invoke the duplex call to absorb a padded message block and XOR the message block with the extracted output to get the ciphertext block.

Finalization Invoke the duplex eight times, each with an empty message ϵ absorbed. Then invoke the duplex another four times, before each of which a 32-bit output is squeezed. Concatenate the four 32-bit output blocks to form the 128-bit tag.

For Subterranean-deck, the input of Subterranean-deck consists of a 128-bit key, a string sequence M and the output length. It is noted that the string sequence M here can contain multiple arbitrary-length message strings.

2.4 Toy Examples

To facilitate cryptanalysis, simplified versions of a cipher are useful. As it is impossible to define simpler versions of Subterranean 2.0 with round-reduced permutations, we introduce two size-reduced versions of Subterranean 2.0 as follows.

Subterranean 2.0 uses a prime-sized state to avoid the existence of exploitable symmetries. Therefore, the state size b of a toy example also needs to be prime but smaller than 257. In addition, the factor d used in the π step should have a large order in $\mathbb{Z}_b^*$ and the order should be a multiple of the extraction rate. With these in mind, we choose two primes 193 and 97 and let d be a generator of $\mathbb{Z}_b^*$, resulting in two toy examples: Subterranean-m and Subterranean-s whose parameters are summarized in Table 2.

Table 2: Toy examples of Subterranean 2.0

Version	State size	Key size	d	Extraction rate	Output Z_i
Subterranean 2.0	257	128	12	32	$s_{12^{4i}} + s_{-12^{4i}}$
Subterranean-m	193	96	15	32	$s_{15^{3i}} + s_{-15^{3i}}$
Subterranean-s	97	48	15	16	$s_{15^{3i}} + s_{-15^{3i}}$

3 Properties of Subterranean 2.0 and Three Attack Scenarios

In this section, we highlight several important properties of Subterranean 2.0 and relate them to three attack scenarios.

Subterranean 2.0 is a duplex-based construction and uses bit-oriented operations that allow good performance in hardware implementation. Besides, the following properties are interesting in the attacker's point of view.

Property 1. Subterranean 2.0 employs an extremely simple permutation in the duplex call. The permutation has only one round and the round function has algebraic degree only 2. Additionally, the round function operates at bit level and allows a minimum of sub-structures by using a prime-sized state. That is to say, the round function is of weak alignment.

Property 2. Subterranean 2.0 squeezes output blocks in a way similar to a stream cipher. Specifically, it outputs 32 bits as the keystream iteratively before each duplex call. In Subterranean-SAE, when the message is fixed as a constant, it can be seen as a keystream generator in the message processing phase.

Property 3. Subterranean-SAE processes the nonce with multiple duplex calls. Subterranean-SAE does not load the nonce into its initial state. Because of its small state size, Subterranean-SAE has to absorb the nonce with multiple duplex calls and the number of the duplex calls is 5.

Attack scenario 1: keystream biases. When considering Property 1 and Property 2 together, one may ask: are the keystreams truly random? Therefore, the bias in the keystream would be of great concern. Recently, exploitable biases using linear combinations of output bits were found in the authenticated encryption scheme MORUS [SSS+19]. It is important to known if this will happen to Subterranean 2.0.

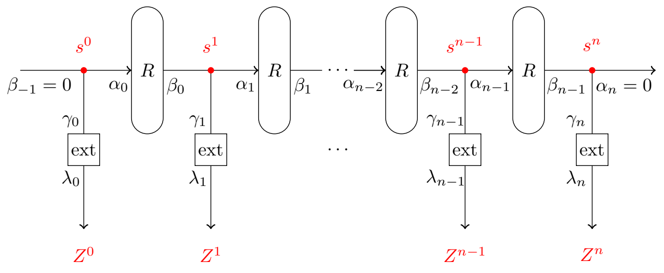

Figure 4: Linear trails for keystream bias evaluation

To investigate the bias of keystreams, it is to find a sequence of linear masks $(\lambda_0, \cdots, \lambda_n)$ for the output blocks Z^i, as illustrated in Figure 4, such that

$$\sum_{i=0}^{n} \lambda_i Z^i$$

is biased. If such a sequence of masks can be found, a distinguisher can be mounted on the target. Here, the same tools for linear cryptanalysis of block ciphers can be applied with the beginning and end being set inactive, *i.e.*, $\beta_{-1} = 0$, $\alpha_n = 0$ as shown in Figure 4. In the middle, the propagation of linear masks must be compatible with each operation. Summing all approximations:

$$\begin{aligned} &\gamma_i s^i + \lambda_i Z^i, \quad 0 \le i \le n, \\ &\alpha_i s^i + \beta_i s^{i+1}, \ 0 \le i \le n-1, \end{aligned}$$

we will have $\sum_{i=0}^{n} \lambda_i Z^i$, and for Subterranean 2.0, the correlation of keystreams Z^i is equivalent to the correlation of the round functions involved as the extraction function is linear.

The designers kept the above attack in mind while designing Subterranean 2.0 and let the output Z be extracted from special state bits to prevent any bias in three consecutive output blocks. It is believed that using four or more output blocks eliminates measurable bias in Z. Any evidence that approves or disapproves of such a belief would be interesting to the community.

Attack scenario 2: state collisions. A similar cryptanalysis in the differential case would be state collision attacks. As illustrated in Figure 5, the difference of the internal state is introduced by an input difference ΔX^0 (through the nonce, AD or the message), and cancelled out by ΔX^n after n rounds. Such an attack is called "LOCAL attack" which was proposed by Khovratovich and Rechberger [KR13] and independently found by Wu *et al.* [WWH+13] against ALE [BMR+13].

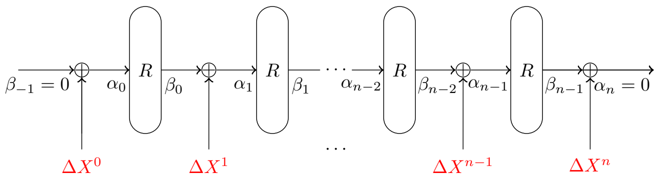

Figure 5: Differential trails for state collisions

The state collision may cause forgery attacks. Suppose the internal difference is introduced by the associated data AD and there exists such a differential trail with high probability p. Then a forgery attack can be mounted in the following way.

Let N, $A_0||\cdots||A_n$ and M be the nonce, AD and message to be forged. The attacker respects nonces and queries $(N, A_0 \oplus \Delta X^0||\cdots||A_n \oplus \Delta X^n, M)$ to the encryption oracle to get the 128-bit tag T. Then, T is a valid tag for $(N, A_0||\cdots||A_n, M)$ with probability p. The forgery attack succeeds if $p > 2^{128}$.

As the nonce is processed in multiple duplex calls, it might be possible to find state collision during the nonce processing phase. If the state collision happens during this phase and there are more bits of nonce to be absorbed after the collision, *i.e.*, $(N_1||N_2, A, M)$ and $(N_1'||N_2, A, M)$ lead to a state collision and thus to the same tag T, then for any A' and M', the attacker can make forgeries by using a new N_2.

In spite of the importance of the security requirement for resisting state collision attacks, such a differential analysis is missing, either in the specification of Subterranean 2.0 or in the literature.

Attack scenario 3: state recovery in the nonce-misuse setting. Subterranean-SAE takes a nonce as input and strongly relies on nonce uniqueness for security. Even though no security claim was made in the nonce-misuse setting, it is believed by the designers in [DMMR20] that the state recovery attack is non-trivial, as quoted below.

> *In nonce-misuse scenario's or when unwrapping invalid cryptograms returns more information than a simple error, we make no security claims and an attacker may even be able to reconstruct the secret state. Nevertheless we believe that this would probably a non-trivial effort, both in attack complexity as in ingenuity.* .

Recall Property 1 that Subterranean 2.0 uses the one-round permutation with algebraic 2 in the duplex call. In the setting that a nonce can be used more than once, one may inject a difference $\Delta\overline{M}^i$ at s^i in the message processing phase as shown in Figure 6, one will obtain some linear relations of the state difference Δs^{i+1} through the output difference ΔZ^{i+2} as each output bit is the sum of two internal bits. More importantly, Δs^{i+1} is linear in bits of s^i due to Fact 1 for quadratic Boolean functions. Therefore, ΔZ^{i+2} will be linear in s^i, and thus some bits of s^i will be leaked by observing such one-round differentials.

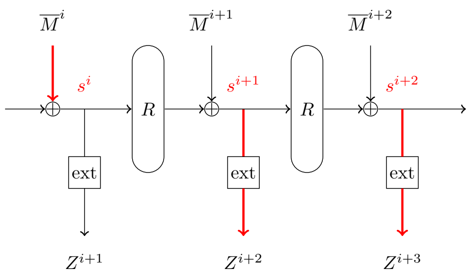

Figure 6: Notations for state recovery in the nonce-misuse setting

Fact 1. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function with algebraic degree 2. Then the difference Δf can be expressed linearly by the input bits.*

Example 1. *Let $f : \mathbb{F}_2^2 \to \mathbb{F}_2$ and $f = x_0 \cdot x_1$. Suppose the input difference is given as $(\Delta x_0, \Delta x_1)$. Then $\Delta f = x_0 \cdot x_1 + (x_0 + \Delta x_0) \cdot (x_1 + \Delta x_1) = \Delta x_1 \cdot x_0 + \Delta x_0 \cdot x_1 + \Delta x_0 \cdot \Delta x_1$.*

Even though Subterranean-SAE aims for use cases where nonce uniqueness can be guaranteed, it would be interesting to know what the complexity of state recovery would be when nonce uniqueness is lost.

In the following two sections, the three potential attacks pointed out here will be investigated. Section 4 looks into differential and linear cryptanalysis regarding keystream biases and state collisions respectively, and Section 5 examines state recovery attack in the nonce-misuse setting.

4 Differential and Linear Analysis Tailored for Keystream Biases and State Collisions

Since the work by Mouha *et al.* [MWGP11], various automatic tools have been developed for searching differential and linear trails, such as [MP13, SHS+13, KLT15]. In this section, we first conduct a succinct analysis of the non-linear layer χ of the round function and choose an appropriate tool for Subterranean 2.0. Then we present the results we obtained and discuss the impact on Subterranean-deck and Subterranean-SAE.

4.1 Analysis of χ

Subterranean 2.0 uses bit-wise operations. In particular, in the χ step, for $0 \le i < b$,

$$x_i \leftarrow x_i + x_{i+1} \cdot x_{i+2} + x_{i+2}.$$

It would not be difficult to build a model for counting the number of active ANDs. However, unlike S-box based ciphers where the number of active S-boxes determines the upper bound of differential/linear probability, the number of active ANDs provides not much information for Subterranean 2.0. The reason is the dependency between ANDs.

Take the linear case for instance. Each AND operation $x_i \cdot x_{i+1}$ can be approximated by $0, x_i, x_{i+1}$ or $x_i + x_{i+1}$ with correlation 2^{-1} (the bias is 2^{-2}). Let us see Example 2 where two ANDs share an input bit.

Example 2. *Let $f(x_0, x_1, x_2) = x_0 \cdot x_1 + x_1 \cdot x_2 + L(x_0, x_1, x_2) = x_0 \cdot x_1 + x_1 \cdot x_2 + u \cdot x_0 + v \cdot x_1 + w \cdot x_2$ be a Boolean function and $u, v, w \in \mathbb{F}_2$ are constants. If $u = w$, then $Cor(f) = 2^{-1}$; otherwise, $Cor(f) = 0$.*

There are half cases of $L(x_0, x_1, x_2)$ leading to zero correlation of f because $x_0 \cdot x_1 + x_1 \cdot x_2$ cannot be approximated by the linear part when $u \neq w$. As $x_0 \cdot x_1 + x_1 \cdot x_2 = x_1 \cdot (x_0 + x_2)$, the correlation of f is 2^{-1} instead of 2^{-2} when the correlation is not zero. Therefore, handling the dependency between ANDs is the key point of tools for searching linear trails for AND-based ciphers. For differential trails, the dependency also has a similar effect.

In previous works like [SHS+17, SSS+19, SSS+20], trails for AND-based ciphers are searched by treating the AND operations as independent ones in their tools. When the trails are obtained, the validity is checked separately by dedicated procedures. This strategy is useful when the trails are sparse. However, it is not the case for Subterranean 2.0 when keystream biases or state collisions are considered. The experiments show that the trails obtained with this strategy are almost invalid. What's worse, such inexact models are unable to provide reliable bounds of differential/linear probability.

Represent χ as a SIMON-like function We observe that the χ step bears a strong resemblance to SIMON's round function. SIMON [BSS+13] is a family of lightweight block ciphers and follows the Feistel construction. Its round function has the following form

$$S^{-\alpha}(x) \odot S^{-\beta}(x) \oplus S^{-\gamma}(x),$$

where $S^i(x)$ corresponds to a cyclic right shift of x by i bits, $\odot$ and $\oplus$ denote the bit-wise AND and XOR operations respectively. We observe that χ can be re-written as a SIMON-like function:

$$x \leftarrow x \oplus S^1(x) \odot S^2(x) \oplus S^2(x).$$

Therefore, the techniques and tools in [KLT15] for searching exact differential/linear trails of SIMON serves as a good starting point for differential and linear cryptanalysis of Subterranean 2.0.

4.2 Linear Analysis

In linear cryptanalysis of AND-based ciphers, there are blocks of chained active ANDs where the correlation can be calculated for each block independently. Depending on the number of active ANDs involved in a block, there are two cases which are covered by Lemma 1 and 2. In the case of Subterranean 2.0, $k = 1$ for the two lemmas. When the number n of active ANDs in a block is odd, *i.e.*, $n = 2t - 1, t > 0$, any approximation is valid and the correlation is 2^{-t}. When the number n of active ANDs is even, *i.e.*, $n = 2t, t > 0$, the approximation should satisfy a condition *cond* as stated in Lemma 2. This is a one-bit condition and if it holds, the correlation is 2^{-t}. In other words, given a random approximation for an even block, it is valid with probability $\frac{1}{2}$. In search of linear trails, it is the key point to make sure this condition holds for all even blocks. Without this condition being imposed, the obtained linear trail will be invalid with high chance when the trail is dense.

Lemma 1. *Let $f(x) = x_0x_k + x_kx_{2k} + \cdots + x_{(2t-2)k}x_{(2t-1)k} + L(x_0, x_k, \cdots, x_{(2t-1)k})$ be a Boolean function where L is linear and $t > 0$. Then $Cor(f)$ is 2^{-t}.*

Proof. The quadratic part of $f(x)$ can be re-written as

$$x_k(x_0 + x_{2k}) + x_{3k}(x_{2k} + x_{4k}) + \cdots + x_{(2t-3)k}(x_{(2t-4)k} + x_{(2t-2)k}) + x_{(2t-2)k}x_{(2t-1)k}.$$

Apply the following transformation:

$$\begin{aligned} y_{(2j-1)k} &= x_{(2j-1)k}, \qquad 1 \leq j \leq t \\ y_{(2j)k} &= x_{(2j)k} + x_{2(j+1)k}, \qquad 0 \leq j \leq t-2 \\ y_{(2t-2)k} &= x_{(2t-2)k}, \end{aligned}$$

which is equivalent to the transformation $\boldsymbol{x} = A\boldsymbol{y}$:

$$\begin{aligned} x_{(2j-1)k} &= y_{(2j-1)k}, \qquad 1 \le j \le t \\ x_{(2j)k} &= \sum_{i=j}^{t} y_{(2i)k}, \qquad 0 \le j \le t-1 \end{aligned}$$

Then one can obtain

$$g(\boldsymbol{y}) = f(A\boldsymbol{y}) = y_0 y_k + y_{2k}y_{3k} + \cdots + y_{2(t-1)k}y_{(2t-1)k} + L'(y_0, y_k, \ldots, y_{(2t-1)k}).$$

Since the quadratic terms of g contains all $y_{jk}, 0 \le j \le 2t-1$, $Cor(g) = 2^{-t}$. Therefore, $Cor(f) = 2^{-t}$, as

$$Cor(g) = \frac{1}{2^{2t}} \sum_{\boldsymbol{y} \in \mathbb{F}_2^{2t}} (-1)^{g(\boldsymbol{y})} = \frac{1}{2^{2t}} \sum_{\boldsymbol{y} \in \mathbb{F}_2^{2t}} (-1)^{f(A\boldsymbol{y})} = \frac{1}{2^{2t}} \sum_{\boldsymbol{y} \in \mathbb{F}_2^{2t}} (-1)^{f(\boldsymbol{y})} = Cor(f).$$

□

Lemma 2. *Let* $f(x) = x_0x_k + x_kx_{2k} + \cdots + x_{(2t-2)k}x_{(2t-1)k} + x_{(2t-1)k}x_{2tk} + L_0(x_0, x_{2k} \cdots, x_{2tk}) + L_1(x_k, x_{3k}, \cdots, x_{(2t-1)k})$ *be a Boolean function where* L_0, L_1 *are linear and* $t > 0$. *Let* ***cond*** *be:* L_0 *contains a even number of terms. Then* $Cor(f)$ *is* 2^{-t} *if* ***cond*** *holds and 0 otherwise.*

Proof. The quadratic part of $f(x)$ can be re-written as

$$x_k(x_0 + x_{2k}) + x_{3k}(x_{2k} + x_{4k}) + \cdots + x_{(2t-1)k}(x_{(2t-2)k} + x_{(2t)k})$$

Apply the following transformation:

$$\begin{aligned} y_{(2j-1)k} &= x_{(2j-1)k}, \qquad 1 \le j \le t \\ y_{(2j)k} &= x_{(2j)k} + x_{2(j+1)k}, \qquad 0 \le j \le t-1 \\ y_{(2t)k} &= x_{(2t)k}, \end{aligned}$$

which is equivalent to the transformation $\boldsymbol{x} = A\boldsymbol{y}$:

$$x_{(2j-1)k} = y_{(2j-1)k}, \qquad 1 \le j \le t-1 \tag{1}$$

$$x_{(2j)k} = \sum_{i=j}^{t} y_{(2i)k}, \qquad 0 \le j \le t \tag{2}$$

Then one can obtain

$$g(\boldsymbol{y}) = f(A\boldsymbol{y}) = y_0 y_k + y_{2k}y_{3k} + \cdots + y_{2(t-1)k}y_{(2t-1)k} + L(y_0, y_k, \ldots, y_{(2t)k}).$$

Obviously, $Cor(g) = 0$ if $L(y_0, y_k, \cdots, y_{(2t)k})$ contains the term $y_{(2t)k}$, otherwise $Cor(g) = 2^{-t}$. And $L(y_0, y_k, \cdots, y_{(2t)k})$ has the term $y_{(2t)k}$ if and only if $L_0(x_0, x_{2k}, \ldots, x_{(2t)k})$ contains a odd number of terms according to Eq. (2). □

Technically, for an even block with $2t, t > 0$ chained active ANDs, it requires $t+1$ iterations to check the condition *cond*. Hence, the longer an even block is, the more time-consuming for the checking. As the state size of Subterranean 2.0 is 257 which is relatively large when compared to block ciphers like SIMON, the length of even block can reach 256 theoretically. In order to speed up the search for linear trails of *Subterranean*2.0, it would be useful to identify a tighter upper bound of block length l for each round. This can be done as follows when the range of correlation or the target correlation is given.

1. For round r, set the target correlataion C, time limit D and set the block length as state size, *i.e.*, $l = b$
   (a) For all possible positions for a block with l chained ANDs:
      i. Set the l ANDs active. If a solution is found or the searching time exceeds D, exit.
   (b) $l = l - 1$ and go to (a).

We then propose two models:

1. Set l to a reasonable value for all rounds, *e.g.*, $l = 6$. This model is used for searching linear trails with good correlations.
2. For each round, set l to the upper bound found by the above procedure. This model is used for providing tighter lower bounds of correlation of linear trails.

We apply these two models to Subterranean 2.0 and the obtained results are summarized in Table 3. By design, there is no valid linear trail in 3, 2, 2 keystream blocks for Subterranean 2.0, Subterranean-m, and Subterranean-s respectively. For Subterranean-s and Subterranean-m, the optimal linear trails using 3 keystream blocks are found and the correlations are 2^{-30} and 2^{-51}, lower than 2^{-24} and 2^{-48} respectively. For Subterranean 2.0, the linear trail using 4 keystream blocks we find so far has correlation 2^{-96} and there does not exist any such trail with correlation higher or equal to 2^{-49}. When one more keystream block is taken into consideration, no better linear trails are found for Subterranean-s, while it becomes infeasible to search for optimal linear trails for Subterranean-m on a desktop.

Table 3: Result of searching linear biases of keystreams

Version	$(\|s\|, \|K\|)$	$\|Z^i\|$	$\#Z^i$	$-\log_2(Cor)$
Subterranean-SAE	(257,128)	32	4	$(49, 96]$
Subterranean-m	(193,96)	32	3	51
Subterranean-s	(97,48)	16	3	30

4.3 Differential Analysis

In differential cryptanalysis of Subterranean 2.0, we slightly adapt Theorem 1 from [KLT15] and then apply it to Subterranean 2.0.

Theorem 1 ([KLT15])**.** *Let $f(x) = S^1(x) \odot x$ be a Boolean function on $\mathbb{F}_2^n$. The probability that difference α goes to difference β through f is*

$$\Pr(\alpha \xrightarrow{f} \beta) = \begin{cases} 2^{-n+1} & \alpha = \mathbf{1} \text{ and } wt(\beta) \equiv 0 \bmod 2, \\ 2^{-wt(vb+db)} & \alpha \neq \mathbf{1} \text{ and } \beta \odot \overline{vb} = \mathbf{0} \text{ and } (S^1(\beta) \oplus \beta) \odot db = \mathbf{0}, \\ 0 & \text{otherwise,} \end{cases}$$

where $vb = S^1(\alpha) \vee \alpha$, $db = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha)$ and $wt(x)$ is the Hamming weight of x.

The original Theorem 1 considers bit vector x of an even number of bits. When the state size is odd, the condition for the first case should be adapted to $wt(\beta) \equiv 1$.

Even though the search for differential trails is more efficient than the search for linear trails as described in the previous subsection, we can only prove the best differential trail with 4 input blocks has probability p where $-\log_2(p) \in (96, 180]$. However, interesting results are found in the toy versions. With 3 input blocks, the optimal differential probability of Subterranean-m and Subterranean-s are 2^{-104} and 2^{-58} which are lower than 2^{-96} and 2^{-48} respectively. With one more input blocks, *i.e.*, 4 blocks, better differential trails with probability 2^{-90} and 2^{-41} are found, as shown in Table 8, 7, which are higher than 2^{-96} and 2^{-48} respectively. Our results are summarized in Table 4.

Table 4: Result of searching differential trails for state collisions

Version	$(\lvert s\rvert, \lvert K\rvert)$	$\lvert\Delta\overline{M}^i\rvert$	$\#\Delta\overline{M}^i$	$-\log_2(p)$	$\#\Delta\overline{M}^i$	$-\log_2(p)$
Subterranean-SAE	(257,128)	32+1	4	$(108, 180]$	-	-
Subterranean-m	(193,96)	32+1	3	104	4	≤ 90
Subterranean-s	(97,48)	16+1	3	58	4	41

4.4 Impact on Subterranean-deck and Subterranean-SAE

As between extractions or injections, there is only one round, there does not exist any clustering effect in the differential/linear analysis. Thus the security of Subterranean 2.0 against the linear attack and the state collision attack is known from optimal differential/linear trails.

Bias of keystream. For both Subterranean-deck and Subterranean-SAE, the security is claimed against attackers that are limited to 2^{96} data blocks. Thus a useful linear trail should have correlation higher than 2^{-48}. Our results show that with a minimal number of keystream blocks, such linear trails do not exist for Subterranean 2.0 as well as two toy versions. When we increase the number of keystream blocks by 1, there are no better linear trails in Subterranean-s, which gives some confidence that there might be no better linear trails as well for Subterranean 2.0. In short, our results partially support the designers' claim on the security against linear cryptanalysis.

State collisions. State collisions can be used for probabilistic forgeries as long as the differential probability $p > 2^{-|K|}$ when the tag length is the same as the key length. That is, the forgery attack is not constrained by the data limit. Searching differential trails for Subterranean 2.0 is hard, while good differential trails can be found for the two toy versions, with which forgery attacks can be mounted. According to the results on the toy versions, the resistance of Subterranean 2.0 against state collision attacks might be weaker than its resistance against linear attacks.

5 Key Recovery of Subterranean-SAE in the Nonce-misuse Setting

In this section, it is shown that a practical state recovery attack can be mounted with only 88 32-bit blocks and 20 repetitions of nonce by one-round differential analysis.

5.1 One-round Differential Analysis

In the duplex call of Subterranean 2.0, a single-round permutation is used. As the round function has algebraic degree only 2, the output difference of the round function will be linear in the input. So is the difference of the following keystream block. Let us explain the idea with an example as follows.

Example 3. *Suppose one bit difference is injected at position* 1 *of* s^i *(see Figure 6). After one round, the bits at positions* $[0, 64, 85, 107, 150, 171, 192, 214, 235]$ *of* s^{i+1} *have difference* $[s^i_2, s^i_2, s^i_2, s^i_0+1, 1, s^i_0+1, s^i_0+1, 1, 1]$ *and there is no difference at other positions. From the extraction, we have* $\Delta Z^{i+2}_8 = \Delta s^{i+1}_{64} + \Delta s^{i+1}_{193} = s^i_2$*. Thus obtain one state bit* s^i_2 *by observing* ΔZ^{i+2}*.*

This means, in the message processing phase, if a difference is injected at s^i, some state bits of s^i can be recovered by observing the output difference after one round. We call this *one-round differential* of Subterranean 2.0. As can be seen that the recovered bits are among the neighbouring bits of the injected difference. For Subterranean-SAE, the number of bit positions for injection is 32. Further analysis shows that only 41 neighbouring bits can be recovered by one-round differentials.

5.2 Nested One-round Differential Analysis

To enlarge the number of state bits that can be recovered, we propose a *nested one-round differential analysis* which exploits the output difference in two consecutive rounds. The core idea is that injecting difference at s^i will lead to differences of s^{i+1} at positions that may fall outside the set of 32 injection positions. Therefore, besides injecting difference through the input block, we can also utilize the difference generated by the previous round, and thus treat the previous round as a difference injector.

It is known that the difference after two rounds is not linear in the input bits anymore. However, by our nested one-round differential analysis, some bits of the internal state can still be recovered as long as the input difference to the second round is sparse. Next, we illustrate the nested one-round differential by Example 4.

Example 4. *Suppose one bit difference is injected at position* 1 *of* s^i *(see Figure 6). Treat the second round independently with input difference* $[s^i_2, s^i_2, s^i_2, s^i_0+1, 1, 1, s^i_0+1, s^i_0+1, 1, 1]$ *at positions* $[0, 64, 85, 107, 150, 171, 192, 214, 235]$ *based on Example 3. By observing the difference of the output block after the second round* ΔZ^{i+3}*, retrieve relations between* s^{i+1}*,* s^i_0, s^i_2 *through* ΔZ^{i+3}*, and select the linear ones which are:*

$$\begin{aligned}
\Delta Z^{i+3}_1 &= s^0_2,\\
\Delta Z^{i+3}_3 &= s^0_0 + 1,\\
\Delta Z^{i+3}_8 &= s^0_2,\\
\Delta Z^{i+3}_{12} &= s^1_{234} + 1,\\
\Delta Z^{i+3}_{13} &= s^1_{149} + 1,\\
\Delta Z^{i+3}_{14} &= s^0_2,\\
\Delta Z^{i+3}_{16} &= s^0_0 + 1,\\
\Delta Z^{i+3}_{22} &= s^1_{213} + 1,\\
\Delta Z^{i+3}_{23} &= s^1_{215}.
\end{aligned}$$

Therefore, 6 bits: $s^0_0, s^0_2, s^1_{149}, s^1_{213}, s^1_{215}, s^1_{234}$ *can be recovered.*

5.3 Key Recovery

In our attack, we utilize 9 types of difference injections No. $1 \sim 9$ as listed in Table 5. Using 19 injections of difference in total, 131 bits information of s^1 and 128 bits information of s^2 can be known, as illustrated in Table 6. With this information, the full state s^1 can be recovered as follows.

Guess another 26 bits of s^1, as listed in the last row of Table 5, then all bits of s^2 can be expressed in 257-131-26 = 100 unknowns and there remain 26 quadratic terms composed of these unknowns after χ. When the 26 quadratic terms are treated as independent unknowns, there will be 100+26 unknown. As 128 bits of s^2 are known, a system of 128 linear equations in 126 unknowns can be constructed and solved easily. The time complexity of recovering the full s^1 is 2^{26}.

Table 5: Difference injection and state recovery

No.	Pos. of s^i with difference	Recovered bits	#Recovered bits
1	15, 213, 223, 211, 134, 128, 35, 234, 70, 190, 184, 111, 165, 169, 11, 4, 22	$s^i_5, s^i_{12}, s^i_{16}, s^i_{21}, s^i_{34}, s^i_{69}, s^i_{71}, s^i_{110}, s^i_{112}, s^i_{133}, s^i_{129}, s^i_{135}, s^i_{164}, s^i_{166}, s^i_{168}, s^i_{185}, s^i_{189}, s^i_{191}, s^i_{210}, s^i_{212}, s^i_{214}, s^i_{224}, s^i_{233}, s^i_{235}, s^i_3 + s^i_{10}$, and 5 extra bits $s^i_{241}, s^i_{223}, s^i_{128}, s^i_{68}, s^i_{22}$	30 bits of s^i
2	137, 140, 30, 225, 197, 189, 95, 2, 256, 249	$s^i_1, s^i_3, s^i_{29}, s^i_{94}, s^i_{96}, s^i_{136}, s^i_{139}, s^i_{190}, s^i_{198}, s^i_{196}, s^i_{226}, s^i_{250}, s^i_{255}, s^{i+1}_{169} + s^{i+1}_{172}$ and 4 extra bits $s^i_{256}, s^i_{121}, s^i_{67}, s^i_2$	17 bits of s^i, 1 bit of s^{i+1}
3	136, 176, 1	$s^i_{177}, s^i_2, s^i_{137}, s^i_0, s^i_{175}, s^{i+1}_{234}, s^{i+1}_{181}, s^{i+1}_{215}, s^{i+1}_{213}, s^{i+1}_{160}, s^{i+1}_{162}, s^{i+1}_{13} + s^{i+1}_{249}$ and 3 extra bits $s^{i+1}_{23}, s^{i+1}_{44}, s^{i+1}_{95}$	5 bits of s^i, 10 bits of s^{i+1}
4	137, 64	$s^i_{63}, s^i_{138}, s^{i+1}_{246}, s^{i+1}_{92}, s^{i+1}_{76}, s^{i+1}_{248}, s^{i+1}_{154}, s^{i+1}_{74}, s^{i+1}_{55}, s^{i+1}_{156}$ and 2 extra bits $s^{i+1}_{11}, s^{i+1}_{165}$	2 bits of s^i, 10 bits of s^{i+1}
5	4,22	$s^i_{23}, s^{i+1}_{172}, s^{i+1}_{170}, s^{i+1}_{24}, s^{i+1}_{149}, s^{i+1}_{87}, s^{i+1}_{217}, s^{i+1}_{85}$, and 1 extra bit s^i_{234}	2 bits of s^i, 7 bits of s^{i+1}
6	11, 140, 241	$s^i_{242}, s^i_{240}, s^{i+1}_{171}, s^{i+1}_{192}, s^{i+1}_{107}, s^{i+1}_{194}, s^{i+1}_{254}, s^{i+1}_{182}$ and 2 extra bits s^i_{15}, s^i_{17}	4 bits of s^i, 6 bits of s^{i+1}
7	17,70,35,165	$s^i_{36}, s^{i+1}_{66}, s^{i+1}_{109}, s^{i+1}_{238}, s^{i+1}_{79}, s^{i+1}_{141}, s^{i+1}_{143}, s^{i+1}_{47} + s^{i+1}_{221}, s^{i+1}_{49} + s^{i+1}_{219}$	1 bit of s^i, 8 bits of s^{i+1}
8	211, 95, 169	$s^i_{170}, s^{i+1}_{201}, s^{i+1}_{116}, s^{i+1}_{40}, s^{i+1}_{229}, s^{i+1}_{163}, s^{i+1}_{114}, s^{i+1}_{104}, s^{i+1}_{123}$ and 1 extra bit s^{i+1}_{134}	1 bit of s^i, 9 bits of s^{i+1}
9	256,189, 223	$s^i_{222}, s^{i+1}_{103}, s^{i+1}_{193}, s^{i+1}_{108}, s^{i+1}_{106}, s^{i+1}_{105}, s^{i+1}_{81}, s^i_0 \cdot s^{i+1}_{43} + s^{i+1}_{39}$ and 3 extra bits $s^i_{35}, s^{i+1}_{64}, s^{i+1}_{176}$	2 bits of s^i, 9 bits of s^{i+1}

Recover the key Once the whole state s^1 is obtained, the 128-bit key can be recovered by a guess-and-determine procedure as in [LIM19]. First, with s^1, the state after injecting K^4 can be computed. As K^4 is unknown, only 225 bits of the state are known before the injection. Then, guess 32 bits of K^1 and 3 bits of K^2 at positions $[2, 136, 189]$ so that the state after injecting K^3 are linear in the remaining 29 bits of K^2 and the full 32 bits of K^3. Hence, the 225 known bits before injecting K^4 are quadratic in these 61 key bits. A detailed analysis shows that the expressions of the 225 known bits contain at most 128 quadratic terms. Again if we treat these 128 quadratic terms as independent unknowns, then there will be a system of 61+128 unknowns and 225 linear equations. The solution of the system provides information of (K^1, K^2, K^3). When (K^1, K^2, K^3) is obtained, recovering K^4 is trivial. As a result, recovering the key from s^1 takes a time complexity of 2^{35}. In summary, the key can be recovered practically if the same nonce repeats 20 times.

Relation to the extraction function. In the squeezing phase, Subterranean 2.0 outputs a block of 32 bits, each of which is the sum of two state bits: $Z_i = s_{12^{4i}} + s_{-12^{4i}}$, for $0 \le i < 32$. Instead of outputting state bits directly, this extraction function is meant to frustrate state recovery attacks [FNR18] in the nonce respected setting. In our one-round differential analysis, this extraction function allows more state bits involved in the output block and thus more state bits can be recovered. For example, if we set $Z_i = s_{12^{4i}}$, for $0 \le i < 32$, type 1 injection of difference will lead to a recovery of 17 bits versus 30 bits

Table 6: State recovery with 19 injections of difference under the nonce-misuse setting

	Recovered bits of s^1	Recovered bits of s^2
No. 3 $\sim$ 9 at s^0	59 bits: $s^1_{234}, s^1_{181}, s^1_{215}, s^1_{213}, s^1_{160}, s^1_{162}, s^1_{13} + s^1_{249}, s^1_{23}, s^1_{44}, s^1_{95}, s^1_{246}, s^1_{92}, s^1_{76}, s^1_{248}, s^1_{154}, s^1_{74}, s^1_{55}, s^1_{156}, s^1_{11}, s^1_{165}, s^1_{172}, s^1_{170}, s^1_{24}, s^1_{149}, s^1_{87}, s^1_{217}, s^1_{85}, s^1_{171}, s^1_{192}, s^1_{107}, s^1_{194}, s^1_{254}, s^1_{182}, s^1_{66}, s^1_{109}, s^1_{238}, s^1_{79}, s^1_{141}, s^1_{143}, s^1_{47} + s^1_{221}, s^1_{49} + s^1_{219}, s^1_{201}, s^1_{116}, s^1_{40}, s^1_{229}, s^1_{163}, s^1_{114}, s^1_{104}, s^1_{123}, s^1_{134}, s^1_{103}, s^1_{193}, s^1_{108}, s^1_{106}, s^1_{105}, s^1_{81}, s^1_{64}, s^1_{176}, s^i_0 \cdot s^1_{43} + s^1_{39}$ (as s^0_0 can be known)	
No. 1 $\sim$ 9 at s^1	60 bits: $s^1_5, s^1_{12}, s^1_{16}, s^1_{21}, s^1_{34}, s^1_{69}, s^1_{71}, s^1_{110}, s^1_{112}, s^1_{133}, s^1_{129}, s^1_{135}, s^1_{166}, s^1_{168}, s^1_{185}, s^1_{189}, s^1_{191}, s^1_{210}, s^1_{212}, s^1_{214}, s^1_{224}, s^1_{233}, s^1_{235}, s^1_3 + s^1_{10}, s^1_{241}, s^1_{223}, s^1_{128}, s^1_{68}, s^1_{22}, (s^1_{164}), s^1_1, s^1_3, s^1_{29}, s^1_{94}, s^1_{96}, s^1_{136}, s^1_{139}, s^1_{190}, s^1_{198}, s^1_{196}, s^1_{226}, s^1_{250}, s^1_{255}, s^1_{256}, s^1_{121}, s^1_{67}, s^1_2 s^1_{177}, s^1_2, s^1_{137}, s^1_0, s^1_{175}, s^1_{63}, s^1_{138}(s^1_{234}), (s^1_{23}), s^1_{242}, s^1_{240}, s^1_{15}, s^1_{17} s^1_{36}, (s^1_{170}), s^1_{222}, s^1_{35}$	60 bits: $s^2_{169} + s^2_{172}, s^2_{234}, s^2_{181}, s^2_{215}, s^2_{213}, s^2_{160}, s^2_{162}, s^2_{13} + s^2_{249}, s^2_{23}, s^2_{44}, s^2_{95}, s^2_{246}, s^2_{92}, s^2_{76}, s^2_{248}, s^2_{154}, s^2_{74}, s^2_{55}, s^2_{156}, s^2_{11}, s^2_{165}, s^2_{172}, s^2_{170}, s^2_{24}, s^2_{149}, s^2_{87}, s^2_{217}, s^2_{85}, s^2_{171}, s^2_{192}, s^2_{107}, s^2_{194}, s^2_{254}, s^2_{182}, s^2_{66}, s^2_{109}, s^2_{238}, s^2_{79}, s^2_{141}, s^2_{143}, s^2_{47} + s^2_{221}, s^2_{49} + s^2_{219}, s^2_{201}, s^2_{116}, s^2_{40}, s^2_{229}, s^2_{163}, s^2_{114}, s^2_{104}, s^2_{123}, s^2_{134}, s^2_{103}, s^2_{193}, s^2_{108}, s^2_{106}, s^2_{105}, s^2_{81}, s^2_{64}, s^2_{176}, s^1_0 \cdot s^2_{43} + s^2_{39}$
No. 1 $\sim$ 3 at s^2		52 bits: $s^2_5, s^2_{12}, s^2_{16}, s^2_{21}, s^2_{34}, s^2_{69}, s^2_{71}, s^2_{110}, s^2_{112}, s^2_{133}, s^2_{129}, s^2_{135}, s^2_{164}, s^2_{166}, s^2_{168}, s^2_{185}, s^2_{189}, s^2_{191}, s^2_{210}, s^2_{212}, s^2_{214}, s^2_{224}, s^2_{233}, s^2_{235}, s^2_3 + s^2_{10}, s^2_{241}, s^2_{223}, s^2_{128}, s^2_{68}, s^2_{22}, s^2_1, s^2_3, s^2_{29}, s^2_{94}, s^2_{96}, s^2_{136}, s^2_{139}, s^2_{190}, s^2_{198}, s^2_{196}, s^2_{226}, s^2_{250}, s^2_{255}, s^2_{256}, s^2_{121}, s^2_{67}, s^2_2, s^2_{177}, s^2_2, s^2_{137}, s^2_0, s^2_{175}$
In total	1 additional bit from No. 9 injection: $s^1_{31} = s^2_{76} + s^2_{201} + s^2_{196} + s^1_{94} + s^1_{226} * s^2_{189} + s^1_{226} + 1 + \Delta Z^2_5 + \Delta Z^2_{15}$. Thus, 120 bits plus 11 remaining extraction equations	112 bits plus 16 remaining extraction equations
Guess 26 bits: $s^1_{49}, s^1_{47}, s^1_8, s^1_{184}, s^1_{60}, s^1_{43}, s^1_{111}, s^1_{19}, s^1_{26}, s^1_{51}, s^1_{53}, s^1_{57}, s^1_{62}, s^1_{83}, s^1_{89}, s^1_{98}, s^1_{100}, s^1_{118}, s^1_{125}, s^1_{131}, s^1_{152}, s^1_{158}, s^1_{179}, s^1_{203}, s^1_{205}, s^1_{207}$, and there remains only 26 quadratic terms in the expressions of s^2.		

under the original extraction and 20 state bits can be recovered with ordinary one-round differential analysis versus 41 state bits under the original extraction. Note that our one-round differential analysis requires a nonce-misuse setting.

Comparison to Liu *et al.*'s work. In [LIM19], Liu *et al.* presented a practical state-recovery attack in the nonce-misuse setting with 2^{13} 32-bit blocks based on conditional cube analysis. It was exploited that when the condition holds, the sum of over a set of outputs will be zero. Liu *et al.* mainly utilized a 2-dimensional set to recover one bit, which means 4 repetitions of nonce are required for retrieving 1 state bit. On the contrary, as many as 30 state bits can be recovered with 2 repetitions of nonce by a one-round differential. Therefore, the data complexity is much lower in our one-round differential analysis.

6 Concluding Remarks

In this paper, we investigated the "one-round permutation" in various phases of Subterranean 2.0 and identified three related attack scenarios that deserve further analysis: keystream biases in the keyed squeezing phase, state collisions in the keyed absorbing phase, and one-round differentials in the message processing phase when a nonce is reused.

To carry out a study on the security in the first two attack scenarios, it is necessary to search for differential/linear trails under special constraints. First, we proposed two toy versions of Subterranean 2.0: Subterranean-m and Subterranean-s, to understand Subterranean 2.0 with easier effort. Besides, we observed a resemblance between the non-linear layer of the round function of Subterranean 2.0 and SIMON's round function. Such resemblance offers a good starting point for differential/linear analysis of Subterranean 2.0. On top of the existing tool for searching differential/linear trails of SIMON, we built our own models for Subterranean 2.0. Our results showed that there do not exist linear trails under the constraint of data limit imposed in the design specification with a minimal number of output blocks. This partially supports the designers' claim on the security of Subterranean 2.0 against the linear attack. Regarding state collisions in keyed modes, we found useful differential trails of the toy versions with which forgery attacks can be mounted successfully. However, due to the time-consuming search for differential trails of Subterranean 2.0, its security against the state collision attack in keyed modes still remains an open question.

Finally, we observed that one-round differentials allow to recover state bits in the nonce-misuse setting. In order to recover a sufficient number of state bits, we further proposed nested one-round differentials where a one-round differential is prepended to another, acting as a difference injector. As a result, a practical state recovery attack can be achieved with only 20 repetitions of the nonce and 88 blocks of data. Our analysis shows that Subterranean-like constructions with quadratic one-round permutation must be used carefully in practice as the security crashes when nonce uniqueness is lost.

References

[BDPV11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

[BDPVA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic Sponge Functions. *Submission to NIST (Round 3)*, 2011. http://sponge.noekeon.org/CSF-0.1.pdf.

[BMR+13] Andrey Bogdanov, Florian Mendel, Francesco Regazzoni, Vincent Rijmen, and Elmar Tischhauser. ALE: AES-Based Lightweight Authenticated Encryption. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 447–466. Springer, 2013.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.

[CDGP93] Luc J. M. Claesen, Joan Daemen, Mark Genoe, and G. Peeters. Subterranean: A 600 Mbit/Sec Cryptographic VLSI Chip. In *Proceedings 1993 International Conference on Computer Design: VLSI in Computers & Processors, ICCD '93, Cambridge, MA, USA, October 3-6, 1993*, pages 610–613. IEEE Computer Society, 1993.

[DMMR20] Joan Daemen, Pedro Maat Costa Massolino, Alireza Mehrdad, and Yann Rotella. The Subterranean 2.0 Cipher Suite. *IACR Trans. Symmetric Cryptol.*, 2020(S1):262–294, 2020.

[FNR18] Thomas Fuhr, María Naya-Plasencia, and Yann Rotella. State-Recovery Attacks on Modified Ketje Jr. *IACR Trans. Symmetric Cryptol.*, 2018(1):29–56, 2018.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON Block Cipher Family. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 161–185. Springer, 2015.

[KR13] Dmitry Khovratovich and Christian Rechberger. The LOCAL Attack: Cryptanalysis of the Authenticated Encryption Scheme ALE. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 174–184. Springer, 2013.

[LIM19] Fukang Liu, Takanori Isobe, and Willi Meier. Cube-Based Cryptanalysis of Subterranean-SAE. *IACR Trans. Symmetric Cryptol.*, 2019(4):192–222, 2019.

[MP13] Nicky Mouha and Bart Preneel. A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2013:328, 2013.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

[Nat19] National Institute of Standards and Technology. Lightweight Cryptography (LWC) Standardization Project, 2019. https://csrc.nist.gov/projects/lightweight-cryptography.

[SHS+13] Siwei Sun, Lei Hu, Ling Song, Yonghong Xie, and Peng Wang. Automatic Security Evaluation of Block Ciphers with S-bP Structures Against Related-Key Differential Attacks. In Dongdai Lin, Shouhuai Xu, and Moti Yung, editors, *Information Security and Cryptology - 9th International Conference, Inscrypt 2013, Guangzhou, China, November 27-30, 2013, Revised Selected Papers*, volume 8567 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2013.

[SHS+17] Danping Shi, Lei Hu, Siwei Sun, Ling Song, Kexin Qiao, and Xiaoshuang Ma. Improved linear (hull) cryptanalysis of round-reduced versions of SIMON. *Sci. China Inf. Sci.*, 60(3):39101:1–39101:3, 2017.

[SSS+19] Danping Shi, Siwei Sun, Yu Sasaki, Chaoyun Li, and Lei Hu. Correlation of Quadratic Boolean Functions: Cryptanalysis of All Versions of Full \mathsf MORUS. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 180–209. Springer, 2019.

[SSS+20] Dhiman Saha, Yu Sasaki, Danping Shi, Ferdinand Sibleyras, Siwei Sun, and Yingjie Zhang. On the Security Margin of TinyJAMBU with Refined Differential and Linear Cryptanalysis. To appear in IACR Trans. Symmetric Cryptol. 2020(3), 2020.

[WWH+13] Shengbao Wu, Hongjun Wu, Tao Huang, Mingsheng Wang, and Wenling Wu. Leaked-State-Forgery Attack against the Authenticated Encryption Algorithm ALE. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 377–404. Springer, 2013.

A Differential Trails

Table 7: Differential trail of Subterranean-s with probability 2^{-41} for state collisions

Round i	Difference		$-\log_2(p_i)$
0	ΔZ	0x000000020000804060000000	9
	α	0x000000020000804060000000	
	β	0x000000020000804048000000	
	$\pi \circ \theta(\beta)$	0x000060600200084000142010 0	
1	ΔZ	0x000000020000084004800110 0	21
	α	0x000060620200000004942100 0	
	β	0x00004842020000000 4d639400	
	$\pi \circ \theta(\beta)$	0x10000080000108002 0d040586	
2	ΔZ	0x1000008000010800208000102	11
	α	0x0000000000000000005040484	
	β	0x0000000000000000004040484	
	$\pi \circ \theta(\beta)$	0x0000000000010800000080102	
3	ΔZ	0x0000000000010800000080102	
	α	0x0000000000000000000000000	

Table 8: Differential trail of Subterranean-m with probability 2^{-90} for state collisions

Round i	Difference		$-\log_2(p_i)$
0	ΔZ	0x0000800000000000000000000400000000000000000000000	4
	α	0x0000800000000000000000000400000000000000000000000	
	β	0x0000800000000000000000000600000000000000000000000	
	$\pi \circ \theta(\beta)$	0x0000820000000000000400204000000004000040001000000 40	
1	ΔZ	0x0000820000000000000220060000000003000040000288009 02	30
	α	0x0000000000000000000620264000000007000000001288009 42	
	β	0x100000000000000000005b03650000000058000000001 32a009f3	
	$\pi \circ \theta(\beta)$	0x0086043021020e0200020000040000020 0d2840c1402ae112	
2	ΔZ	0x14000000100c000200020000040000020000000088088 00b02	56
	α	0x14860430310e0e0000000000000000000 0d28404948aaea10	
	β	0x11a505282d8d08000000000000000000000009be707f1c8c9f1c	
	$\pi \circ \theta(\beta)$	0x0400000000008000020000030040000410000400800180020 2	
3	ΔZ	0x0400000000008000020000030040000410000400800180020 2	
	α	0x00000000000000000000000000000000000000000000000000	