Cryptanalysis of MORUS

Tomer Ashur[1], Maria Eichlseder[2], Martin M. Lauridsen, Gaëtan Leurent[3], Brice Minaud[4], Yann Rotella[3], Yu Sasaki[5], and Benoît Viguier[6]

[1] imec-COSIC, KU Leuven, Belgium
[2] Graz University of Technology, Austria
[3] Inria, France
[4] Royal Holloway University of London, United Kingdom
[5] NTT, Japan
[6] Radboud University, Netherlands

tomer.ashur@esat.kuleuven.be, maria.eichlseder@iaik.tugraz.at, mail@martinlauridsen.info, gaetan.leurent@inria.fr, brice.minaud@gmail.com, yann.rotella@inria.fr, b.viguier@science.ru.nl, sasaki.yu@lab.ntt.co.jp

Abstract. MORUS is a high-performance authenticated encryption algorithm submitted to the CAESAR competition, and recently selected as a finalist. There are three versions of MORUS: MORUS-640 with a 128-bit key, and MORUS-1280 with 128-bit or 256-bit keys. For all versions the security claim for confidentiality matches the key size. In this paper, we analyze the components of this algorithm (initialization, state update and tag generation), and report several results.

As our main result, we present a linear correlation in the keystream of full MORUS, which can be used to distinguish its output from random and to recover some plaintext bits in the broadcast setting. For MORUS-1280, the correlation is 2^{-76}, which can be exploited after around 2^{152} encryptions, less than what would be expected for a 256-bit secure cipher. For MORUS-640, the same attack results in a correlation of 2^{-73}, which does not violate the security claims of the cipher.

To identify this correlation, we make use of rotational invariants in MORUS using linear masks that are invariant by word-rotations of the state. This motivates us to introduce single-word versions of MORUS called MiniMORUS, which simplifies the analysis. The attack has been implemented and verified on MiniMORUS, where it yields a correlation of 2^{-16}.

We also study reduced versions of the initialization and finalization of MORUS, aiming to evaluate the security margin of these components. We show a forgery attack when finalization is reduced from 10 steps to 3, and a key-recovery attack in the nonce-misuse setting when initialization is reduced from 16 steps to 10. These additional results do not threaten the full MORUS, but studying all aspects of the design is useful to understand its strengths and weaknesses.

Keywords: MORUS, CAESAR, Authenticated Encryption, Nonce Respecting, Linear Cryptanalysis, Confidentiality.

1 Introduction

Authenticated Encryption (AE) schemes combine the functionality of symmetric encryption schemes and message authentication codes. Based on a shared secret key K, they encrypt a plaintext message M to a ciphertext C and authentication tag T in order to protect both the confidentiality and the authenticity of M. Most modern authenticated encryption algorithms are nonce-based schemes with associated data (AEAD), where (C, T) additionally depends on a unique nonce N (or initialization value IV) and optional associated metadata A. One of the most prominent standardized AEAD designs is AES-GCM [13,8], which is widely deployed in protocols such as TLS (since v1.2).

To address the growing need for modern authenticated encryption designs for different application scenarios, the CAESAR competition was launched in 2013 [4]. The goal of this competition is to select a final portfolio of AEAD designs for three different use-cases: (1) lightweight hardware characteristics, (2) high-speed software performance, and (3) robustness. The competition attracted 57 first-round submissions, 7 of which were recently selected as finalists in the fourth selection round.

MORUS is one of the three finalists for use-case (2), together with OCB and AEGIS. This family of authenticated ciphers by Wu and Huang [19] provides three main variants: MORUS-640 with a 128-bit key and MORUS-1280 with either a 128-bit or a 256-bit key. The design approach is reminiscent of classical stream cipher designs and continuously updates a relatively large state with a few fast operations. MORUS can be efficiently implemented in both software and hardware; in particular, the designers claim that the software performance even surpasses AES-GCM implementations using Intel's AES-NI instructions, and that MORUS is the fastest authenticated cipher not using AES-NI [19].

Related Work. In the MORUS submission document, the designers discuss the security of MORUS against several attacks, including algebraic, differential, and guess-and-determine attacks. The main focus is on differential properties, and not many details are given for other attack vectors. In third-party analysis, Mileva et al. [14] propose a distinguisher in the nonce-reuse setting and practically evaluate the differential behaviour of toy variants of MORUS. Shi et al. [17] analyze the differential properties of the finalization reduced to 2 out of 10 steps, but find no attacks. Dwivedi et al. [6] discuss the applicability of SAT solvers for state recovery, but the resulting complexity of 2^{370} for MORUS-640 is well beyond the security claim. Dwivedi et al. [7] also propose key-recovery attacks for MORUS-1280 if initialization is reduced to 3.6 out of 16 steps, and discuss the security of MORUS against internal differentials and rotational cryptanalysis. Salam et al. [16] apply cube attacks to obtain distinguishers for up to 5 out of 16 steps of the initialization of MORUS-1280 with negligible complexity. Additionally, Kales et al. [9] and Vaudenay and Vizár [18] independently propose state-recovery and forgery attacks on MORUS in a nonce-misuse setting with negligible data and time complexities.

Finally, a keystream correlation similar in nature to our main attack was uncovered by Minaud [15] on the authenticated cipher AEGIS [21,20], another CAESAR finalist. AEGIS shares the same overall structure as MORUS, but uses a very different state update function, based on the parallel application of AES rounds, rather than the shift/AND/XOR operations used in MORUS. Similar to our attack, the approach in [15] is to build a linear trail linking ciphertext bits, while canceling the contribution of inner state bits. How the trail is built depends primarily on the state update function, and how it lends itself to linear cryptanalysis. Because the state update function differs significantly between AEGIS and MORUS, the process used to build the trail is also quite different.

Our Contributions. Our main contribution is a keystream distinguisher on full MORUS-1280, built from linear approximations of its core `StateUpdate` function. In addition, we provide results for round-reduced MORUS, targeting both the initialization or finalization phases of the cipher.

In more detail, our main result is a linear approximation [11,12] linking plaintext and ciphertext bits spanning five consecutive encryption blocks. Moreover, the correlation does not depend on the secret key of the cipher. In principle, this property could be used as a known-plaintext distinguisher, or to recover unknown bits of a plaintext encrypted a large number of times. For MORUS-1280 with 256-bit keys, the linear correlation is 2^{-76} and can be exploited using about 2^{152} encrypted blocks.

To the best of our knowledge, this is the first attack on full MORUS in the nonce-respecting setting. We note that rekeying does not prevent the attack: the biases are independent of the secret encryption key and nonce, and can be exploited for plaintext recovery as long as a given plaintext segment is encrypted sufficiently often, regardless of whether each encryption uses a different key. A notable feature of the linear trail underpinning our attack is also that it does not depend on the values of rotation constants: a very similar trail would exist for most choices of round constants.

To obtain this result, we propose a simplified abstraction of MORUS, called MiniMORUS. MiniMORUS takes advantage of certain rotational invariants in MORUS and simplifies the description and analysis of the attack. We then show how the attack can be extended from MiniMORUS to the real MORUS. To confirm the validity of our analysis, we practically verified the correlation of the full linear trail for MiniMORUS, as well as the correlation of trail fragments for the full MORUS. Our analysis is also backed by a symbolic evaluation of the full trail equation and its correlation on all variants of MORUS.

In addition to the previous attack on full MORUS, we provide two secondary results: (1) we analyze the security of MORUS against forgery attacks with round-reduced finalization; and (2) we analyze its security against key recovery in a nonce-misuse setting, with round-reduced initialization. While this extra analysis does not threaten full MORUS, it complements the main result to provide a better overall understanding of the security of MORUS. More precisely, we present a forgery attack for round-reduced MORUS-1280 with success prob-

ability 2^{-88} for a 128-bit tag if the finalization is reduced to 3 out of 10 steps. This nonce-respecting attack is based on a differential analysis of the padding rule. The second result targets round-reduced initialization with 10 out of 16 steps, and extends a state-recovery attack (which can be mounted e.g. in a nonce-misuse setting) into a key-recovery attack.

Outline. This paper is organized as follows. We first provide a brief description of MORUS in Section 2. In Section 3, we introduce MiniMORUS, an abstraction of MORUS based on a certain class of rotational invariants. We analyze this simplified scheme in Section 4 and provide a ciphertext-only linear approximation with a weight of 16. We then extend our result to the full scheme in Section 5, showing a correlation in the keystream over 5 steps, and discuss the implications of our observation for the security of MORUS in Section 6. In Section 7, we present our results on the security of MORUS with round-reduced initialization (in a nonce-misuse setting) or finalization. We conclude in Section 8.

2 Preliminaries

MORUS is a family of authenticated ciphers designed by Wu and Huang [19]. An instance of MORUS is parametrized by a secret key K. During encryption, it takes as input a plaintext message M, a nonce N, and possibly some associated data A, and outputs a ciphertext C together with an authentication tag T. In this section, we provide a brief description of MORUS and introduce the notation for linear approximations.

2.1 Specification of MORUS

The MORUS family supports two internal state sizes: 640 and 1280 bits, referred to as MORUS-640 and MORUS-1280, respectively. Three parameter sets are recommended: MORUS-640 supports 128-bit keys and MORUS-1280 supports either 128-bit or 256-bit keys. The tag size is 128 bits or shorter. The designers strongly recommend using a 128-bit tag. With a 128-bit tag, integrity is claimed up to 128 bits and confidentiality is claimed up to the number of key bits (Table 1).

State. The internal state of MORUS is composed of five q-bit *registers* S_i, $i \in \{0, 1, 2, 3, 4\}$, where $q = 128$ for MORUS-640 and $q = 256$ for MORUS-1280.

Table 1: Security goals of MORUS.

	Confidentiality (bits)	Integrity (bits)
MORUS-640-128	128	128
MORUS-1280-128	128	128
MORUS-1280-256	256	128

Table 2: Rotation constants b_i for $\lll_w$ and b'_i for $\lll$ in round i of MORUS.

	Bit-wise rotation $\lll_w$					Word-wise rotation $\lll$				
	b_0	b_1	b_2	b_3	b_4	b'_0	b'_1	b'_2	b'_3	b'_4
MORUS-640	5	31	7	22	13	32	64	96	64	32
MORUS-1280	13	46	38	7	4	64	128	192	128	64

The internal state of MORUS may be represented as $S_0\|S_1\|S_2\|S_3\|S_4$. Registers are themselves divided into four $q/4$-bit *words*. Throughout the paper, we denote the word size by $w = q/4$, i.e., $w = 32$ for MORUS-640 and $w = 64$ for MORUS-1280.

The encryption process of MORUS consists of four parts: initialization, associated data processing, encryption, and finalization. During the initialization phase, the value of the state is initialized using a key and nonce. The associated data and the plaintext are then processed block by block. Then the internal state undergoes the finalization phase, which outputs the authentication tag.

Every part of this process relies on iterating the `StateUpdate` function at the core of MORUS. Each call to the `StateUpdate` function is called a step. The internal state at step t is denoted by $S_0^t\|S_1^t\|S_2^t\|S_3^t\|S_4^t$, where $t = -16$ before the initialization and $t = 0$ after the initialization.

The `StateUpdate` Function. `StateUpdate` takes as input the internal state $S^t = S_0^t\|S_1^t\|S_2^t\|S_3^t\|S_4^t$ and an additional q-bit value m^t (recall that q is the size of a register), and outputs an updated internal state.

`StateUpdate` is composed of 5 rounds with similar operations. The additional input m^t is used in rounds 2 to 5, but not in round 1. Each round uses the bit-wise rotation (left circular shift) operation inside word, denoted $\lll_w$ in the following and `Rotl_xxx_yy` in the design document. It divides a q-bit register value into 4 words of $w = q/4$ bits, and performs a rotation on each w-bit word. The bit-wise rotation constants b_i for round i are defined in Table 2. Additionally, each round uses rotations on a whole q-bit register by a multiple of the word size, denoted $\lll$ in the following and `<<<` in the design document. The word-wise rotation constants b'_i are also listed in Table 2.

$S^{t+1} \leftarrow \texttt{StateUpdate}(S^t, m^t)$ is defined as follows, where $\cdot$ denotes bit-wise AND, $\oplus$ is bit-wise XOR, and m_i is defined depending on the context:

$$\begin{aligned}
&\text{Round 1:} && S_0^{t+1} \leftarrow (S_0^t \oplus (S_1^t \cdot S_2^t) \oplus S_3^t) \lll_w b_0, && S_3^t \leftarrow S_3^t \lll b'_0.\\
&\text{Round 2:} && S_1^{t+1} \leftarrow (S_1^t \oplus (S_2^t \cdot S_3^t) \oplus S_4^t \oplus m_i) \lll_w b_1, && S_4^t \leftarrow S_4^t \lll b'_1.\\
&\text{Round 3:} && S_2^{t+1} \leftarrow (S_2^t \oplus (S_3^t \cdot S_4^t) \oplus S_0^t \oplus m_i) \lll_w b_2, && S_0^t \leftarrow S_0^t \lll b'_2.\\
&\text{Round 4:} && S_3^{t+1} \leftarrow (S_3^t \oplus (S_4^t \cdot S_0^t) \oplus S_1^t \oplus m_i) \lll_w b_3, && S_1^t \leftarrow S_1^t \lll b'_3.\\
&\text{Round 5:} && S_4^{t+1} \leftarrow (S_4^t \oplus (S_0^t \cdot S_1^t) \oplus S_2^t \oplus m_i) \lll_w b_4, && S_2^t \leftarrow S_2^t \lll b'_4.
\end{aligned}$$

Initialization. The initialization of MORUS-640 starts by loading the 128-bit key K_{128} and the 128-bit nonce N_{128} into the state together with constants c_0, c_1:

$$S_0^{-16} = N_{128}, \quad S_1^{-16} = K_{128}, \quad S_2^{-16} = 1^{128}, \quad S_3^{-16} = c_0, \quad S_4^{-16} = c_1.$$

Then, $\texttt{StateUpdate}(S^t, 0)$ is iterated 16 times for $t = -16, -15, \ldots, -1$. Finally, the key is XORed into the state again with $S_1^0 \leftarrow S_1^0 \oplus K_{128}$.

The initialization of MORUS-1280 differs slightly due to the difference in register size and the two possible key sizes, and uses either $K = K_{128} \| K_{128}$ (for MORUS-1280-128) or $K = K_{256}$ (for MORUS-1280-256) to initialize the state:

$$S_0^{-16} = N_{128} \parallel 0^{128}, \quad S_1^{-16} = K, \quad S_2^{-16} = 1^{256}, \quad S_3^{-16} = 0^{256}, \quad S_4^{-16} = c_0 \parallel c_1.$$

After iterating `StateUpdate` 16 times, the state is updated with $S_1^0 \leftarrow S_1^0 \oplus K$.

Associated Data Processing. After initialization, the associated data A is processed in blocks of $q \in \{128, 256\}$ bits. For the padding, if the last associated data block is not a full block, it is padded to q bits with zeroes. If the length of A, denoted by $|A|$, is 0, then the associated data processing phase is skipped; else, the state is updated as

$$S^{t+1} \leftarrow \texttt{StateUpdate}(S^t, A^t) \qquad \text{for } t = 0, 1, \ldots, \lceil |A|/q \rceil - 1.$$

Encryption. Next, the message is processed in blocks M_t of $q \in \{128, 256\}$ bits to update the state and produce the ciphertext blocks C_t. If the last message block is not a full block, a string of 0's is used to pad it to 128 or 256 bits for MORUS-640 and MORUS-1280, respectively, and the padded full block is used to update the state. However, only the partial block is encrypted. Note that if the message length denoted by $|M|$ is 0, encryption is skipped. Let $u = \lceil |A|/q \rceil$ and $v = \lceil |M|/q \rceil$. The following is performed for $t = 0, 1, \ldots, v-1$:

$$\begin{aligned} C^t &\leftarrow M^t \oplus S_0^{u+t} \oplus (S_1^{u+t} \lll b_2') \oplus (S_2^{u+t} \cdot S_3^{u+t}), \\ S^{u+t+1} &\leftarrow \texttt{StateUpdate}(S^{u+t}, M^t). \end{aligned}$$

Finalization. The finalization phase generates the authentication tag T using 10 more `StateUpdate` steps. We only discuss the case where T is not truncated. The associated data length and the message length are used to update the state:

1. $L \leftarrow |A| \parallel |M|$ for MORUS-640 or $L \leftarrow |A| \parallel |M| \parallel 0^{128}$ for MORUS-1280, where $|A|, |M|$ are represented as 64-bit integers.
2. $S_4^{u+v} \leftarrow S_4^{u+v} \oplus S_0^{u+v}$.
3. For $t = u+v, u+v+1, \ldots, u+v+9$, compute $S^{t+1} \leftarrow \texttt{StateUpdate}(S^t, L)$.
4. $T = S_0^{u+v+10} \oplus (S_1^{u+v+10} \lll b_2') \oplus (S_2^{u+v+10} \cdot S_3^{u+v+10})$, or the least significant 128 bits of this value in case of MORUS-1280.

2.2 Notation

In the following, we use linear approximations [11] that hold with probability $\Pr(E) = \frac{1}{2} + \varepsilon$, i.e., they are biased with bias ε. The *correlation* $\mathrm{cor}(E)$ of the approximation and its *weight* $\mathrm{weight}(E)$ are defined as

$$\begin{aligned} \mathrm{cor}(E) &:= 2\Pr(E) - 1 = 2\varepsilon\,, \\ \mathrm{weight}(E) &:= -\log_2 |\,\mathrm{cor}(E)|\,, \end{aligned}$$

where $\log_2()$ denotes logarithm in base 2. By the Piling-Up Lemma, the correlation (resp. weight) of an XOR of independent variables is equal to the product (resp. sum) of their individual correlations (resp. weights) [11].

We also recall the following notation from the previous section, where an *encryption step* refers to one call to the `StateUpdate` function:

C^t : the ciphertext block output during the t-th encryption step.
C^t_j : the j-th bit of C^t, with C^t_0 being the rightmost bit.
S^t_i : the i-th register at the beginning of t-th encryption step.
$S^t_{i,j}$: the j-th bit of S^t_i, with $S^t_{i,0}$ being the rightmost bit.

In the above notation, bit positions are always taken modulo the register size q, i.e., $q = 128$ for MORUS-640 and $q = 256$ for MORUS-1280.

For simplicity, in the remainder, the 0-th encryption step will often denote the encryption step where our linear trail starts. Any encryption step could be chosen for that purpose, as long as at least four more encryption steps follow. In particular the 0-th encryption step from the perspective of the trail does not have to be the first encryption step after initialization.

3 Rotational Invariance and MiniMORUS

To simplify the description of the attack, we assume all plaintext blocks are zero. This assumption will be removed in Section 5.3, where we will show that plaintext bits only contribute linearly to the trail. Recall that the inner state of the cipher consists of five $4w$-bit registers $S_0, \ldots, S_4$, each containing four w-bit words.

3.1 Rotationally Invariant Linear Combinations

We begin with a few observations about the `StateUpdate` function. Besides XOR and AND operations, the `StateUpdate` function uses two types of bit rotations:

1. *bit-wise* rotations perform a circular shift on each word within a register;
2. *word-wise* rotations perform a circular shift on a whole register.

The second type of rotation always shifts registers by a multiple of the word size w. This amounts to a (circular) permutation of the words within the register: for example, if a register contains the words (A, B, C, D), and a word-wise rotation

by w bits to the left is performed, then the register now contains the words (B, C, D, A).

To build our linear trail, we start with a linear combinations of bits within a single register.

Definition 1 (Rotational Invariance). *Recall that w denotes the word size in bits, and $4w$ is the size of a register. A linear combination of the form:*

$$S^t_{i,j(0)} \oplus S^t_{i,j(1)} \oplus \cdots \oplus S^t_{i,j(k)}$$

is said to be rotationally invariant *iff the set of bits $S^t_{i,j(0)}, \ldots, S^t_{i,j(k)}$ is left invariant by a circular shift by w bits; that is, iff:*

$$\{j(i) : i \leq k\} = \{j(i) + w \bmod 4w : i \leq k\}.$$

Example. The following linear combination is rotationally invariant for MORUS-640, i.e. $w = 32$:

$$S^t_{0,0} \oplus S^t_{0,32} \oplus S^t_{0,64} \oplus S^t_{0,96}. \tag{1}$$

This definition naturally extends to a linear combination across multiple registers, and also across ciphertext blocks. The value of such a linear combination is unaffected by word-wise rotations, since those rotations always shift registers by a multiple of the word size. On the other hand, since bit-wise rotations always shift all four words within a register by the same amount, bit-wise rotations preserve the rotational invariance property. Moreover, the XOR of two rotationally invariant linear combinations is also rotationally invariant.

This naturally leads to the idea of building a linear trail using only rotationally invariant linear combinations, which is what we are going to do. As a result, the effect of word-wise rotations can be ignored. Moreover, since all linear combinations we consider are going to be rotationally invariant, they can be described by truncating the linear combination to the first word of a register. Indeed, an equivalent way of saying a linear combination is rotationally invariant, is that it involves the same bits in each word within a register. For example, in the case of (1) above, the four bits involved are the first bit of each of the four words.

3.2 MiniMORUS

In fact, we can go further and consider a reduced version of MORUS where each register contains a single word instead of four. The `StateUpdate` function is unchanged, except for the fact that word-wise rotations are removed: see Figure 1. We call these reduced versions MiniMORUS-640 and MiniMORUS-1280, for MORUS-640 and MORUS-1280 respectively. Since registers in MiniMORUS contain a single word, bit-wise and word-wise rotations are the same operation; for simplicity we write $\lll$ for bit-wise rotations.

Since the trail we are building is relatively complex, we will first describe it on MiniMORUS. We will then extend it to the full MORUS via the previous rotational invariance property.

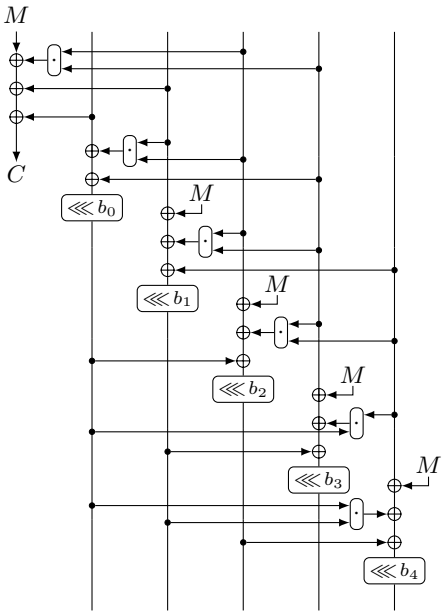


Fig. 1: MiniMORUS state update function.

4 Linear Trail for MiniMORUS

In this section, we describe how we build a trail for MiniMORUS, then compute its correlation and validate the correlation experimentally.

4.1 Overview of the Trail

To build a linear trail for MiniMORUS, we combine the following five trail fragments α_i^t, β_i^t, γ_i^t, δ_i^t, ε_i^t, where the subscript i denotes a bit position, and the superscript t denotes a step number:

- α_i^t approximates (one bit of) state word S_0 using the ciphertext;
- β_i^t approximates S_1 using S_0 and the ciphertext;
- γ_i^t approximates S_4 using two approximations of S_1 in consecutive steps;
- δ_i^t approximates S_2 using two approximations of S_4 in consecutive steps;
- ε_i^t approximates S_0 using two approximations of S_2 in consecutive steps.

The trail fragments are depicted on Figure 2. In all cases except α_i^t, the trail fragment approximates a single AND gate by zero, which holds with probability $3/4$, and hence the trail fragment has weight 1. In the case of α_i^t, two AND gates are involved; however the two gates share an entry in common, and in both cases the other entry also has a linear contribution to the trail, which results in an overall contribution of the form (see [3, Sec. 3.3])

$$x \cdot y \oplus x \cdot z \oplus y \oplus z = (x \oplus 1) \cdot (y \oplus z).$$

As a result, the trail fragment α_i^t also has a weight of 1. Another way of looking at this phenomenon is that the trail holds for two different approximations of the AND gates: the alternative approximation is depicted by a dashed line on Figure 2.

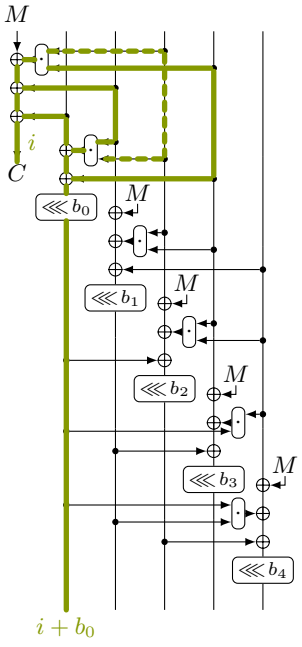
The way we are going to use each trail fragment may be summarized as follows, where in each case, elements to the left of the arrow $\rightarrow$ are used to approximate the element on the right of the arrow:

$$\begin{aligned}
\alpha_i^t &: \quad C_i^t \rightarrow S_{0,i+b_0}^{t+1} \\
\beta_i^t &: \quad C_i^t, S_{0,i}^t \rightarrow S_{1,i}^t \\
\gamma_i^t &: \quad S_{1,i}^t, S_{1,i+b_1}^{t+1} \rightarrow S_{4,i}^t \\
\delta_i^t &: \quad S_{4,i}^t, S_{4,i+b_4}^{t+1} \rightarrow S_{2,i}^{t+1} \\
\varepsilon_i^t &: \quad S_{2,i}^t, S_{2,i+b_2}^{t+1} \rightarrow S_{0,i}^{t+1}.
\end{aligned}$$

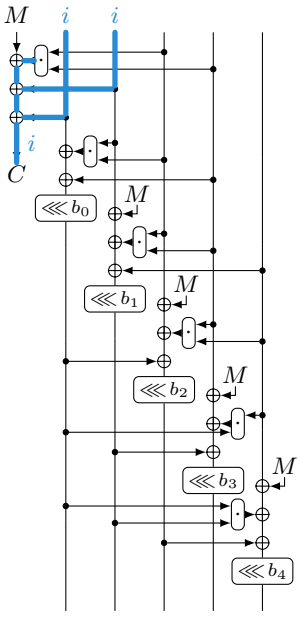
In more detail, the idea is that by using α_i^t, we are able to approximate a bit of S_0 using only a ciphertext bit. By combining α_i^t with $\beta_{i+b_0}^{t+1}$, we are then able to approximate a bit of S_1 (at step $t+1$) using only ciphertext bits from two consecutive steps. Likewise, γ_i^t allows us to "jump" from S_1 to S_4, i.e. by combining α_i^t with β_i^t and γ_i^t with appropriate choices of parameters t and i for each, we are able to approximate one bit of S_4 using only ciphertext bits. Notice however that γ_i^t requires approximating S_1 in two consecutive steps; and so the previous combination requires using α_i^t and β_i^t *twice* at different steps. In the same way, δ_i^t allows us to jump from S_4 to S_2; and ε_i^t allows jumping from S_2 back to S_0. Eventually, we are able to approximate a bit of S_0 using only ciphertext bits via the combination of all trail fragments α_i^t, β_i^t, γ_i^t, δ_i^t, and ε_i^t.

However, the same bit of S_0 can also be approximated directly by using α_i^t at the corresponding step. Thus that bit can be linearly approximated from two different sides: the first approximation uses a combination of all trail fragments, and involves successive approximations of all state registers (except S_3) spanning several encryption steps, as explained in the previous paragraph. The second approximation only involves using α_i^t at the final step reached by the previous trail. By XORing up these two approximations, we are left with only ciphertext bits, spanning five consecutive encryption steps.

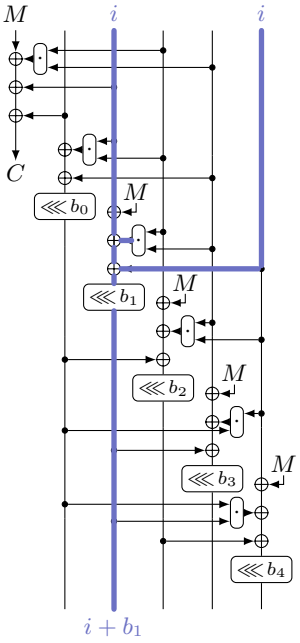
Of course, the overall trail resulting from all of the previous combinations is quite complex, especially since γ_i^t, δ_i^t, and ε_i^t each require two copies of the preceding trail fragment in consecutive steps: that is, ε_i^t requires two approximations of S_2, which requires using δ_i^t twice; and δ_i^t in turn requires using γ_i^t twice, which itself requires using α_i^t and β_i^t twice. Then α_i^t is used one final time to close the trail. The full construction with the exact bit indices for MiniMORUS-640 and MiniMORUS-1280 is illustrated in Figure 3, where the left and right half each show half of the full trail. One may naturally wonder if some components of this trail are in conflict. In particular, products of bits from registers S_2 and S_3 are approximated multiple times, by α_i^t, β_i^t and γ_i^t. To address this concern, and ensure that all approximations along the trail are in fact compatible, we now compute the full trail equation explicitly.



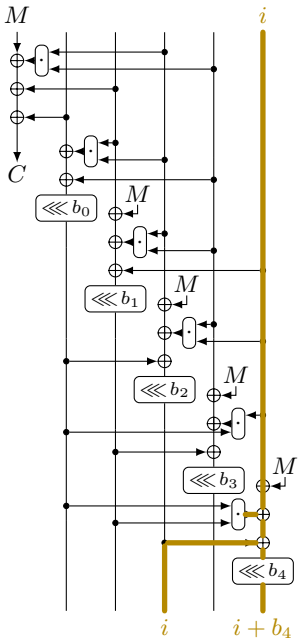
α_i^t: weight 1 (not 2)



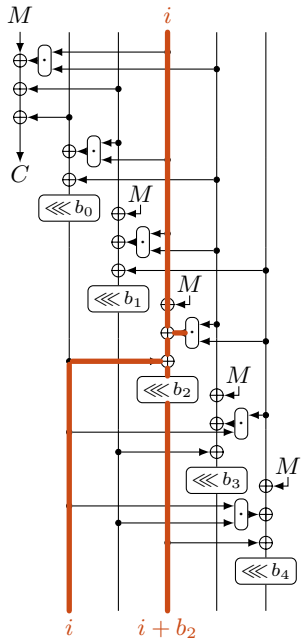
β_i^t: weight 1



γ_i^t: weight 1



δ_i^t: weight 1



ε_i^t: weight 1

MiniMORUS-640:

$$\begin{aligned}
\alpha_i^t &: \quad C_i^t \rightarrow S_{0,i+5}^{t+1} \\
\beta_i^t &: \quad C_i^t, S_{0,i}^t \rightarrow S_{1,i}^t \\
\gamma_i^t &: \quad S_{1,i}^t, S_{1,i+31}^{t+1} \rightarrow S_{4,i}^t \\
\delta_i^t &: \quad S_{4,i}^t, S_{4,i+13}^{t+1} \rightarrow S_{2,i}^{t+1} \\
\varepsilon_i^t &: \quad S_{2,i}^t, S_{2,i+7}^{t+1} \rightarrow S_{0,i}^{t+1}.
\end{aligned}$$

MiniMORUS-1280:

$$\begin{aligned}
\alpha_i^t &: \quad C_i^t \rightarrow S_{0,i+13}^{t+1} \\
\beta_i^t &: \quad C_i^t, S_{0,i}^t \rightarrow S_{1,i}^t \\
\gamma_i^t &: \quad S_{1,i}^t, S_{1,i+46}^{t+1} \rightarrow S_{4,i}^t \\
\delta_i^t &: \quad S_{4,i}^t, S_{4,i+4}^{t+1} \rightarrow S_{2,i}^{t+1} \\
\varepsilon_i^t &: \quad S_{2,i}^t, S_{2,i+38}^{t+1} \rightarrow S_{0,i}^{t+1}.
\end{aligned}$$

MiniMORUS instances

Fig. 2: MiniMORUS linear trail fragments.

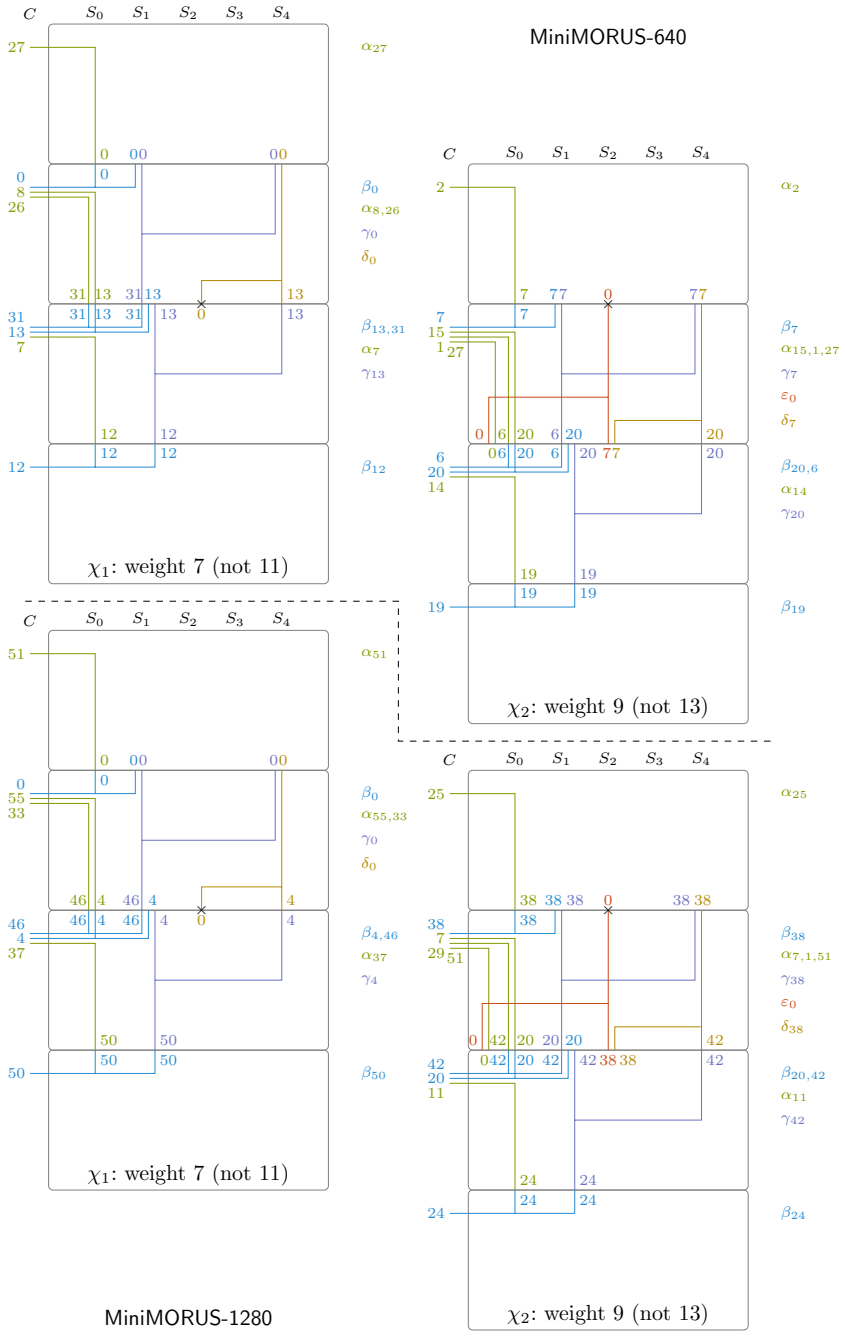


Fig. 3: MiniMORUS: two approximations for $S^2_{2,0}$. Numbers in each diagram denote bit positions used in the linear approximation, i.e. subscripts of $\alpha, \beta, \gamma, \delta$ and ε. χ_1 and χ_2 are two halves of the full trail which we experimentally verify.

4.2 Trail Equation

The equation corresponding to each of the five trail fragments α_i^t, β_i^t, γ_i^t, δ_i^t, ε_i^t may be written explicitly as $\mathbf{A}_i^t$, $\mathbf{B}_i^t$, $\mathbf{C}_i^t$, $\mathbf{D}_i^t$, $\mathbf{E}_i^t$ as follows. For each equation, we write on the left-hand side of the equality the biased linear combination used in the trail; and on the right-hand side, the remainder of the equation, which must have non-zero correlation (in all cases the correlation is 2^{-1}).

$$\begin{aligned}
\mathbf{A}_i^t &: \quad C_i^t \oplus S_{0,i+b_0}^{t+1} = S_{1,i}^t \oplus S_{3,i}^t \oplus S_{1,i}^t \cdot S_{2,i}^t \oplus S_{2,i}^t \cdot S_{3,i}^t \\
\mathbf{B}_i^t &: \quad C_i^t \oplus S_{0,i}^t \oplus S_{1,i}^t = S_{2,i}^t \cdot S_{3,i}^t \\
\mathbf{C}_i^t &: \quad S_{1,i}^t \oplus S_{1,i+b_1}^{t+1} \oplus S_{4,i}^t = S_{2,i}^t \cdot S_{3,i}^t \\
\mathbf{D}_i^t &: \quad S_{4,i}^t \oplus S_{4,i+b_4}^{t+1} \oplus S_{2,i}^{t+1} = S_{0,i}^{t+1} \cdot S_{1,i}^{t+1} \\
\mathbf{E}_i^t &: \quad S_{2,i}^t \oplus S_{2,i+b_2}^{t+1} \oplus S_{0,i}^{t+1} = S_{3,i}^t \cdot S_{4,i}^t
\end{aligned}$$

From an algebraic point of view, building the full trail amounts to adding up copies of the previous equations for various choices of t and i, so that eventually all $S_{y,z}^x$ terms on the left-hand side cancel out. Then we are left with only ciphertext terms on the left-hand side, while the right-hand side consists of a sum of biased expressions. By measuring the correlation of the right-hand side expression, we are then able to determine the correlation of the linear combination of ciphertext bits on the left-hand side. We now set out to do so.

In order to build the equation for the full trail, we start with $\mathbf{E}_0^2$:

$$S_{2,0}^2 \oplus S_{2,b_2}^3 \oplus S_{0,0}^3 = S_{3,0}^2 \cdot S_{4,0}^2.$$

In order to cancel the $S_{0,0}^3$ term on the left-hand side, we add to the equation $\mathbf{A}_{-b_0}^2$ (where the sum of two equations of the form $a = b$ and $c = d$ is defined to be $a + c = b + d$). This yields:

$$\begin{aligned}
&S_{2,0}^2 \oplus S_{2,b_2}^3 \oplus C_{-b_0}^2 \\
&= S_{3,0}^2 \cdot S_{4,0}^2 \oplus S_{1,-b_0}^2 \oplus S_{3,-b_0}^2 \oplus S_{1,-b_0}^2 \cdot S_{2,-b_0}^2 \oplus S_{2,-b_0}^2 \cdot S_{3,-b_0}^2.
\end{aligned}$$

We then need to cancel two terms of the form $S_{2,i}^t$. To do this, we add to the equations $\mathbf{D}_i^t$ for appropriate choices of t and i. This replaces the two $S_{2,i}^t$ terms by four $S_{4,i}^t$ terms. By using equation $\mathbf{B}_i^t$ four times, we can then replace these four $S_{4,i}^t$ terms by eight $S_{1,i}^t$ terms. By applying equation $\mathbf{B}_i^t$ eight times, these eight $S_{1,i}^t$ terms can in turn be replaced by eight $S_{0,i}^t$ terms (and some ciphertext terms). Finally, applying $\mathbf{A}_i^t$ eight times allows to replace these eight $S_{0,i}^t$ terms by only ciphertext bits. Ultimately, for MiniMORUS-1280, this yields the equation:

$$\begin{aligned}
&C_{51}^0 \oplus C_0^1 \oplus C_{25}^1 \oplus C_{33}^1 \oplus C_{55}^1 \oplus C_4^2 \oplus C_7^2 \oplus C_{29}^2 \oplus C_{37}^2 \\
&\oplus C_{38}^2 \oplus C_{46}^2 \oplus C_{51}^2 \oplus C_{11}^3 \oplus C_{20}^3 \oplus C_{42}^3 \oplus C_{50}^3 \oplus C_{24}^4 \\
&= S_{1,51}^0 \cdot S_{2,51}^0 \oplus S_{2,51}^0 \cdot S_{3,51}^0 \oplus S_{1,51}^0 \oplus S_{3,51}^0 \qquad \text{weight 1} \\
&\oplus S_{1,25}^1 \cdot S_{2,25}^1 \oplus S_{2,25}^1 \cdot S_{3,25}^1 \oplus S_{1,25}^1 \oplus S_{3,25}^1 \qquad \text{weight 1}
\end{aligned}$$

$$\oplus\, S^1_{1,33} \cdot S^1_{2,33} \oplus S^1_{2,33} \cdot S^1_{3,33} \oplus S^1_{1,33} \oplus S^1_{3,33} \quad \text{weight } 1$$
$$\oplus\, S^1_{1,55} \cdot S^1_{2,55} \oplus S^1_{2,55} \cdot S^1_{3,55} \oplus S^1_{1,55} \oplus S^1_{3,55} \quad \text{weight } 1$$
$$\oplus\, S^2_{1,7} \cdot S^2_{2,7} \oplus S^2_{2,7} \cdot S^2_{3,7} \oplus S^2_{1,7} \oplus S^2_{3,7} \quad \text{weight } 1$$
$$\oplus\, S^2_{1,29} \cdot S^2_{2,29} \oplus S^2_{2,29} \cdot S^2_{3,29} \oplus S^2_{1,29} \oplus S^2_{3,29} \quad \text{weight } 1$$
$$\oplus\, S^2_{1,37} \cdot S^2_{2,37} \oplus S^2_{2,37} \cdot S^2_{3,37} \oplus S^2_{1,37} \oplus S^2_{3,37} \quad \text{weight } 1$$
$$\oplus\, S^2_{1,51} \cdot S^2_{2,51} \oplus S^2_{2,51} \cdot S^2_{3,51} \oplus S^2_{1,51} \oplus S^2_{3,51} \quad \text{weight } 1$$
$$\oplus\, S^3_{1,11} \cdot S^3_{2,11} \oplus S^3_{2,11} \cdot S^3_{3,11} \oplus S^3_{1,11} \oplus S^3_{3,11} \quad \text{weight } 1$$
$$\oplus\, S^2_{0,0} \cdot S^2_{1,0} \quad \text{weight } 1$$
$$\oplus\, S^2_{2,46} \cdot S^2_{3,46} \quad \text{weight } 1$$
$$\oplus\, S^2_{3,0} \cdot S^2_{4,0} \quad \text{weight } 1$$
$$\oplus\, S^3_{0,38} \cdot S^3_{1,38} \quad \text{weight } 1$$
$$\oplus\, S^3_{2,20} \cdot S^3_{3,20} \quad \text{weight } 1$$
$$\oplus\, S^3_{2,50} \cdot S^3_{3,50} \quad \text{weight } 1$$
$$\oplus\, S^4_{2,24} \cdot S^4_{3,24} \quad \text{weight } 1$$

The equation for MiniMORUS-640 is very similar, and is given in the full version of this paper [2].

4.3 Correlation of the Trail

In the equation for MiniMORUS-1280 from the previous section, each line on the right-hand side of the equality involves distinct $S^t_{i,j}$ terms (in the sense that no two lines share a common term), and each line has a weight of 1. By the Piling-Up Lemma, it follows that if we assume distinct $S^t_{i,j}$ terms to be uniform and independent, then the expression on the right-hand side has a weight of 16. Hence the linear combination of ciphertext bits on the left-hand side has a correlation of 2^{-16}. The same holds for MiniMORUS-640.

The correlation is surprising high. The full trail uses trail fragments ε^t_i, δ^t_i, γ^t_i, β^t_i, and α^t_i, once, twice, 4 times, 8 times, and 9 times, respectively. Since each trail fragment has a weight of 1, this would suggest that the total weight should be $1 + 2 + 4 + 8 + 9 = 24$ rather than 16. However, when combining trail fragments β_i and γ_i, notice that the same AND is computed at the same step between registers S_2 and S_3 (equivalently, notice that the right-hand side of equations $\mathbf{B}^t_i$ and $\mathbf{C}^t_i$ is equal). In both cases it is approximated by zero. When XORing the corresponding equations, these two ANDs cancel each other, which saves two AND gates. Since γ^t_i is used four times in the course of the full trail, this results in saving 8 AND gates overall, which explains why the final correlation is 2^{-16} rather than 2^{-24}.

4.4 Experimental Verification

To confirm that our analysis is correct, we ran experiments on an implementation of MiniMORUS-1280 and MiniMORUS-640. We consider two halves χ_1 and χ_2 of the full trail (depicted on Figure 3), as well as the full trail itself, denoted by χ. In each case, we give the weight predicted by the analysis from the previous section, and the weight measured by our experiments. Results are displayed on Table 3. While our analysis predicts a correlation of 2^{-16}, experiments indicate a slightly better empirical correlation of $2^{-15.5}$ for MORUS-640. The discrepancy of $2^{-0.5}$ probably arises from the fact that register bits across different steps are not completely independent.

The programs we used to verify the bias experimentally are available at:
https://github.com/ildyria/MorusBias

Table 3: Experimental verification of trail correlations.

	Weight	
Approximations for MiniMORUS-640	Predicted	Measured
χ_1 $S_0^{2,2} = C_{27}^0 \oplus C_{0,8,26}^1 \oplus C_{7,13,31}^2 \oplus C_{12}^3$	7	7
χ_2 $S_0^{2,2} = C_2^1 \oplus C_{1,7,15,27}^2 \oplus C_{6,14,20}^3 \oplus C_{19}^4$	9	9
χ $0 = C_{27}^0 \oplus C_{0,2,26,8}^1 \oplus C_{1,13,15,27,31}^2 \oplus C_{6,12,14,20}^3 \oplus C_{19}^4$	16	15.5
Approximations for MiniMORUS-1280		
χ_1 $S_0^{2,2} = C_{51}^0 \oplus C_{0,33,55}^1 \oplus C_{4,37,46}^2 \oplus C_{50}^3$	7	7
χ_2 $S_0^{2,2} = C_{25}^1 \oplus C_{7,29,38,51}^2 \oplus C_{11,20,42}^3 \oplus C_{24}^4$	9	9
χ $0 = C_{51}^0 \oplus C_{0,25,33,55}^1 \oplus C_{4,7,29,37,38,46,51}^2 \oplus C_{11,20,42,50}^3 \oplus C_{24}^4$	16	15.9

5 Trail for Full MORUS

In the previous section, we presented a linear trail for the reduced ciphers MiniMORUS-1280 and MiniMORUS-640. We now turn to the full ciphers MORUS-1280 and MORUS-640.

5.1 Making the Trail Rotationally Invariant

In order to build a trail for the full MORUS, we proceed exactly as we did for MiniMORUS, following the same path down to step and word rotation values, with one difference: in order to move from the one-word registers of MiniMORUS to the four-word registers of full MORUS, we make every term $S_{i,j}^t$ and C_j^t rotationally invariant, in the sense of Section 3. That is, for every $S_{i,j}^t$ (resp. C_j^t) component in every trail fragment and every equation, we expand the term by adding in the terms $S_{i,j+w}^t$, $S_{i,j+2w}^t$, $S_{i,j+3w}^t$ (resp. C_{j+w}^t, C_{j+2w}^t, C_{j+3w}^t), where

as usual w denotes the word size. For example, if $w = 64$ (for MORUS-1280), the term $S^3_{2,0}$ is expanded into:

$$S^3_{2,0} \oplus S^3_{2,64} \oplus S^3_{2,128} \oplus S^3_{2,192}.$$

Thus, translating the trail from one of the MiniMORUS ciphers to the corresponding full MORUS cipher amounts to making every linear combination rotationally invariant—indeed, that was the point of introducing MiniMORUS in the first place. Concretely, in order to build the full trail equation for MORUS, we write rotationally invariant versions of equations $\mathbf{A}^t_i$, $\mathbf{B}^t_i$, $\mathbf{C}^t_i$, $\mathbf{D}^t_i$, $\mathbf{E}^t_i$ from Section 4.2, and then combine them in exactly the same manner as before. This way, the biased linear combination on MiniMORUS-1280 given in Section 4.2, namely:

$$\begin{aligned} &C^0_{51} \oplus C^1_0 \oplus C^1_{25} \oplus C^1_{33} \oplus C^1_{55} \oplus C^2_4 \oplus C^2_7 \oplus C^2_{29} \oplus C^2_{37} \\ &\oplus C^2_{38} \oplus C^2_{46} \oplus C^2_{51} \oplus C^3_{11} \oplus C^3_{20} \oplus C^3_{42} \oplus C^3_{50} \oplus C^4_{24} \end{aligned}$$

ultimately yields the following biased rotationally invariant linear combination on the full MORUS-1280:

$$\begin{aligned} &C^0_{51} \oplus C^0_{115} \oplus C^0_{179} \oplus C^0_{243} \oplus C^1_0 \oplus C^1_{25} \oplus C^1_{33} \oplus C^1_{55} \oplus C^1_{64} \oplus C^1_{89} \\ &\oplus C^1_{97} \oplus C^1_{119} \oplus C^1_{128} \oplus C^1_{153} \oplus C^1_{161} \oplus C^1_{183} \oplus C^1_{192} \oplus C^1_{217} \oplus C^1_{225} \oplus C^1_{247} \\ &\oplus C^2_4 \oplus C^2_7 \oplus C^2_{29} \oplus C^2_{37} \oplus C^2_{38} \oplus C^2_{46} \oplus C^2_{51} \oplus C^2_{68} \oplus C^2_{71} \oplus C^2_{93} \\ &\oplus C^2_{101} \oplus C^2_{102} \oplus C^2_{110} \oplus C^2_{115} \oplus C^2_{132} \oplus C^2_{135} \oplus C^2_{157} \oplus C^2_{165} \oplus C^2_{166} \oplus C^2_{174} \\ &\oplus C^2_{179} \oplus C^2_{196} \oplus C^2_{199} \oplus C^2_{221} \oplus C^2_{229} \oplus C^2_{230} \oplus C^2_{238} \oplus C^2_{243} \oplus C^3_{11} \oplus C^3_{20} \\ &\oplus C^3_{42} \oplus C^3_{50} \oplus C^3_{75} \oplus C^3_{84} \oplus C^3_{106} \oplus C^3_{114} \oplus C^3_{139} \oplus C^3_{148} \oplus C^3_{170} \oplus C^3_{178} \\ &\oplus C^3_{203} \oplus C^3_{212} \oplus C^3_{234} \oplus C^3_{242} \oplus C^4_{24} \oplus C^4_{88} \oplus C^4_{152} \oplus C^4_{216} \end{aligned}$$

We refer the reader to the full version of this paper [2] for the corresponding linear combination on MORUS-640.

5.2 Correlation of the Full Trail

The rotationally invariant trail on full MORUS may be intuitively understood as consisting of four copies of the original trail on MiniMORUS. Indeed, the only difference between full MORUS (for either version of MORUS) and four independent copies of MiniMORUS comes from word-wise rotations, which permute words within a register. But as observed in Section 3, word-wise rotations preserves the rotational invariance property; and so, insofar as we only ever use rotationally invariant linear combinations on all registers along the trail, word-wise rotations have no effect.

Following the previous intuition, one may expect that the weight of the full trail should simply be four times the weight of the corresponding MiniMORUS trail, namely 64 for both MORUS-1280 and MORUS-640. However, reality is a

little more complex, as the full trail does not exactly behave as four copies of the original trail when one considers nonlinear terms.

To understand why that might be the case, assume a nonlinear term $S^0_{2,0} \cdot S^0_{3,0}$ arising from some part of the trail, and another term $S^0_{2,0} \cdot S^0_{3,w}$ arising from a different part of the trail (where w denotes the word size). Then when we XOR the various trail fragments together, in MiniMORUS these two terms are actually equal and will cancel out, since word-wise rotations by multiples of w bits are ignored. However in the real MORUS these terms are of course distinct and do not cancel each other.

In the actual trail for (either version of) full MORUS, this exact situation occurs when combining trail fragments β^t_i and γ^t_i. Indeed, β^t_i requires approximating the term $S^t_{2,i} \cdot S^t_{3,i}$, while γ^t_i requires approximating the term $S^t_{2,i} \cdot S^t_{3,i-w}$ (cf. Figure 4). While in MiniMORUS, these terms cancel out, in the full MORUS, when adding up four copies of the trail to achieve rotational invariance, we end up with the sum:

$$\begin{aligned} &S^t_{2,i} \cdot S^t_{3,i} \oplus S^t_{3,i} \cdot S^t_{2,i+w} \oplus S^t_{2,i+w} \cdot S^t_{3,i+w} \oplus S^t_{3,i+w} \cdot S^t_{2,i+2w} \\ \oplus\ &S^t_{2,i+2w} \cdot S^t_{3,i+2w} \oplus S^t_{3,i+2w} \cdot S^t_{2,i+3w} \oplus S^t_{2,i+3w} \cdot S^t_{3,i+3w} \oplus S^t_{3,i+3w} \cdot S^t_{2,i}. \end{aligned} \quad (2)$$

It may be observed that the products occurring in the equation above involve eight terms forming a ring. The weight of this expression can be computed by brute force, and is equal to 3.

For MORUS-1280, since the trail fragment γ^t_i is used four times, this phenomenon adds a contribution of $4 \cdot 3 = 12$ to the overall weight of the full trail. This results in a total weight of $4 \cdot 16 + 12 = 76$ (recall that the weight of the trail on MiniMORUS-1280 is 16). We have confirmed this by explicitly computing the full trail equation in Appendix A, and evaluating its exact weight like we did for MiniMORUS in Section 4.3. That is, since the equation is quadratic, we may view it as a graph, which we split into connected components; we then compute the weight of each connected component separately by brute force, and then add up the weights of all components per the Piling-Up Lemma. Overall, the full trail equation given in Appendix A yields a weight of 76 for the full trail on MORUS-1280.

In the case of MORUS-640, collisions between rotation constants further complicate the analysis. Specifically, when using trail fragment β^t_i, the term $S^t_{2,i} \cdot S^t_{3,i}$ occurs. As explained previously, a partial collision with the term $S^t_{2,i} \cdot S^t_{3,i-w}$ from trail fragment γ^t_i results in Equation (2). However trail fragment α^t_{i+d} is once used in the course of the full trail with an offset of $d = b_1 + b_4 - b_0 - b_2$ (relative to γ^t_i), which in the case of MORUS-640 is equal to $31 + 13 - 5 - 7 = 0 \bmod 32$. This creates another term $S^t_{2,i} \cdot S^t_{3,i}$, which ultimately destroys one of the four occurrences of Equation (2). Therefore, when computing the full trail equation on MORUS-640, we get that the weight of the trail is 73 (cf. the full version of this paper for the full trail equation for MORUS-640).

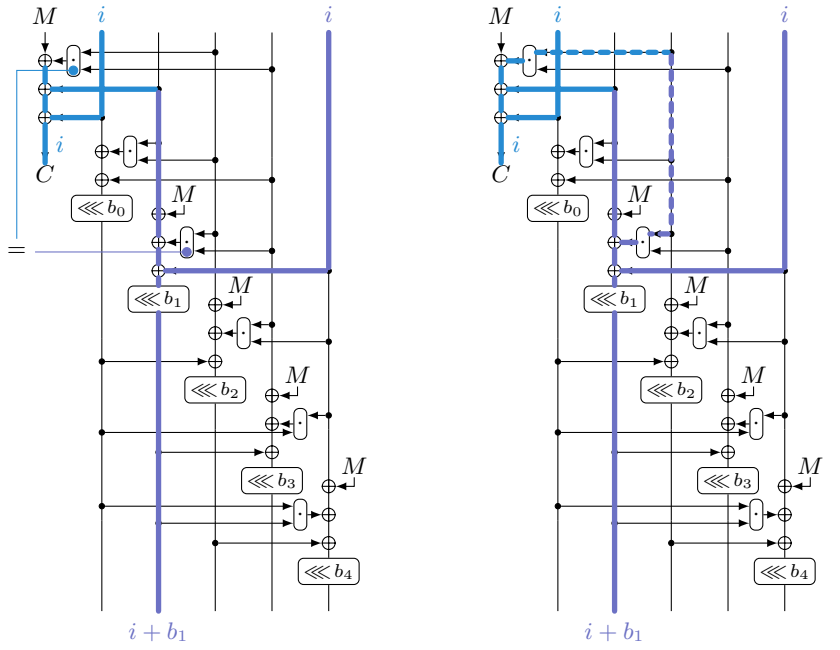


Fig. 4: Weight of $\beta_i^t \oplus \gamma_i^t$ for MiniMORUS and MORUS.

5.3 Taking Variable Plaintext into Account

In our analysis so far, for the sake of simplicity, we have assumed that all plaintext blocks are zero. We now examine what happens if we remove that assumption, and integrate plaintext variables into our analysis. What we show is that plaintext variables only contribute linearly to the trail. In other words, the full trail equation with plaintext variables is equal to the full trail equation with all-zero plaintext XORed with a linear combination of plaintext variables.

To see this, recall that plaintext bits contribute to the encryption process in two ways (cf. Section 2.1):

1. They are added to some bits derived from the state to form the ciphertext.
2. During each encryption step, the `StateUpdate` function adds a plaintext block to every register except S_0.

The effect of Item 1 is that whenever we use a ciphertext bit in our full trail equation, the corresponding plaintext bit also needs to be XORed in. Because ciphertext bits only contribute linearly to the trail equation, this only adds a linear combination of plaintext bits to the equation.

Regarding Item 2, recall that the full trail equation is a linear combination of (the rotationally invariant version of) equations $\mathbf{A}_i^t$, $\mathbf{B}_i^t$, $\mathbf{C}_i^t$, $\mathbf{D}_i^t$, $\mathbf{E}_i^t$ in

Section 4.2. Also observe that in each equation, state bits that are shifted by a bit-wise rotation only contribute linearly. Because plaintext bits are XORed into each register at the same time bit-wise rotation is performed, this implies that plaintext bits resulting from Item 2 also only contribute linearly. In fact in all cases, it so happens that updating the equation to take plaintext variables into account simply involves XORing in the plaintext bit M_i^t.

It may be observed that message blocks in the `StateUpdate` function only contribute linearly to the state, and in that regard play a role similar to key bits in an SPN cipher; and indeed in SPN ciphers, it is the case that key bits contribute linearly to linear trails [11]. In this light the previous result may not be surprising.

In the end, with variable plaintext, our trail yields a biased linear combination of ciphertext bits and plaintext bits. In regards to attacks, this means the situation is effectively the same as with a biased stream cipher: in particular if the plaintext is known we obtain a distinguisher; and if a fixed unknown plaintext is encrypted multiple times (possibly also with some known variable part) then our trail yields a plaintext recovery attack.

6 Discussion

We now discuss the impact of these attacks on the security of MORUS.

Keystream Correlation. We emphasize that the correlation we uncover between plaintext and ciphertext bits is *absolute*, in the sense that it does not depend on the encryption key, or on the nonce. This is the same situation as the keystream correlations in AEGIS [15]. As such, they can be leveraged to mount an attack in the broadcast setting, where the same message is encrypted multiple times with different IVs and potentially different keys [10]. In particular, the broadcast setting appears in practice in man-in-the-browser attacks against HTTPS connections following the BEAST model [5]. In this scenario, an attacker uses Javascript code running in the victim's browser (by tricking the victim to visit a malicious website) to generate a large number of request to a secure website. Because of details of the HTTP protocol, each request includes an authentication token to identify the user, and the attacker can target this token as a repeated plaintext. Concretely, correlations in the RC4 keystream have been exploited in this setting, leading to the recovery of authentication cookies in practice [1].

Data Complexity. The design document of MORUS imposes a limit of 2^{64} encrypted blocks for a given key. However, since our attack is independent of the encryption key, and hence immune to rekeying, this limitation does not apply: all that matters for our attack is that the same plaintext be encrypted enough times.

With the trail presented in this work, the data complexity is clearly out of reach in practice, since exploiting the correlation would require 2^{152} encrypted

blocks for MORUS1280, and 2^{146} encrypted blocks for MORUS640. The data complexity could be slightly lowered by leveraging multilinear cryptanalysis; indeed, the trail holds for any bit shift, and if we assume independence, we could run w copies of the trail in parallel on the same encrypted blocks (recall that w is the word size, and the trail is invariant by rotation by w bits). This would save a factor 2^5 on the data complexity for MORUS640, and 2^6 for MORUS1280; but the resulting complexity is still out of reach.

However, MORUS1280 with a 256-bit key claims a security level of 256 bits for confidentiality, and an attack with complexity 2^{152} violates this claim, even if it is not practical.

Design Considerations. The existence of this trail does hint at some weakness in the design of MORUS. Indeed, a notable feature of the trail is that the values of rotation constants are mostly irrelevant: a similar trail would exist for most choices of the constants. That it is possible to build a trail that ignores rotation constants may be surprising. This would have been prevented by adding a bit-wise rotation to one of the state registers at the input of the ciphertext equation.

7 Analysis on Initialization and Finalization of Reduced MORUS

The bias in the previous sections analysed the encryption part of the MORUS. In this section, for comprehensive security analysis of MORUS, we provide new attacks on reduced version of the initialization and the finalization. We emphasize that the results in this section do not threaten any security claim by the designers. However, we believe that investigating all parts of the design with different approaches from the existing work on MORUS provides a better understanding and will be useful especially when the design will be tweaked in future.

7.1 Forgery with Reduced Finalization

We present forgery attacks on 3 out of 10 steps of MORUS-1280 that claims 128-bit security for integrity. The attack only works for a limited number of steps, while it works in the nonce-respecting setting. As far as we know, this is the first attempt to evaluate integrity of MORUS in the nonce-respecting setting.

Overview. A general strategy for forgery attacks in the nonce-respecting setting is to inject some difference in a message block and propagate it so that it can be canceled by a difference in another message block. However this approach does not work well against MORUS due to its large state size which prevents an attacker from easily controlling the differences in different registers.

Here we focus on the property that the padding for an associated data A and a message M is the zero-padding, hence A and $A' = A\|0^*$ and M and $M' = M\|0$ result in identical states after the associated data processing and the encryption

parts, as long as A, A' and M, M' fit in the same number of blocks. During the finalization, since A, A' (resp. M, M') have different lengths, the corresponding 64-bit values $|A|$ (resp. $|M|$) are different, which appears as $\Delta|A|$ (resp. $\Delta|M|$) during the finalization, and is injected through the message input interface. Our strategy is to propagate this difference to the 128-bit tags T and T' such that their difference ΔT appears with higher probability than 2^{-128}. All in all, the forgery succeeds as long as the desired ΔT is obtained or in other words, the attacker does not have to cancel the state difference, which is the main advantage of attacking the finalization part of the scheme.

Note that if the attacker uses different messages M, M', not only the new tag T' but also new ciphertext C' must be guessed correctly. Because the encryption of MORUS is a simple XOR of the key stream, C' can be easily guessed. For this purpose, the attacker should first query a longer message $M' = M\|0^*$ to obtain C'. Then, C can be obtained by truncating C'.

Differential Trails. Recall that the message input during the finalization of MORUS-1280 is $|A| \parallel |M| \parallel 0^{128}$ where $|A|$ and $|M|$ are 64-bit strings. We set $\Delta|A|$ to be of low Hamming weight, e.g., `0x0000000000000001`. This difference propagates through 3 steps as specified in Table 4.

Recall that each step consists of 5 rounds and the input message is absorbed to the state in rounds 2 to 5. The trail in Table 4 initially does not have any difference and the same continues even after round 1. Differences start to appear from round 2 and they will go through the bitwise-AND operation from round 4. We need to pay 1 bit to control each active AND gate. The probability evaluation for round 15 can be ignored since in this round only S_4 is non-linearly updated, while S_4 is never used for computing the tag. Finally, bitwise-AND in the tag computation is taken into account. Note that the tag is only 128 LSBs, thus the number of active AND gates should be counted only for those bits. As shown in Table 4, we can have a particular tag difference ΔT with probability 2^{-88}. Thus after observing A and corresponding T, $A\|0$ and $(T \oplus \Delta T)$ is a valid pair with probability 2^{-88}.

Remarks. The fact that the S_4 is updated in the last round but is not used in the tag generation implies that the MORUS finalization generally includes unnecessary computations with respect to security. It may be interesting to tweak the design such that the tag can also depend on S_4. Indeed in Table 4, we can observe some jump-up of the probability in the tag computation. This is because the non-linearly involved terms are $S_2 \cdot S_3$, and S_3 that was updated 2 rounds before has a high Hamming weight. In this sense, involving S_4 in non-linear terms of the tag computation imposes more difficulties for the attacker.

7.2 Extending State Recovery to Key Recovery

Kales et al. [9] showed that the internal state of MORUS-640 can be recovered under the nonce-misuse scenario using 2^5 plaintext-ciphertext pairs. As claimed

by [9] the attack is naturally extended to MORUS-1280 though Kales et al. [9] did not demonstrate specific attacks. The recovered state allows the attacker to mount a universal forgery attack under the same nonce. However, the key still cannot be recovered because the key is used both at the beginning and end of the initialization, which prevents the attacker from backtracking the state value to the initial state. In this section, we show that meet-in-the-middle attacks allow the attacker to recover the key faster than exhaustive search for a relatively large number of steps, i.e., 10 out of 16 steps in MORUS-1280.

Overview. We divide the 10 steps of the initialization computation into two subsequent parts F_0 and F_1. (We later set that F_0 is the first 4 steps and F_1 is the last 6 steps.) Let S^{-10} be the initial state value before setting the key, i.e., $S^{-10} = (N \parallel 0^{128}, 0^{256}, 1^{256}, 0^{256}, c_0 \parallel c_1)$. Also let S^0 be 1280-bit state value after the initialization, which is now assumed to be recovered with the nonce-misuse analysis [9]. We then have the following relation.

$$F_1 \circ F_0\big(S^{-10} \oplus (0, K, 0, 0, 0)\big) \oplus (0, K, 0, 0, 0) = S^0.$$

We target the variant MORUS-1280-128, where $K = K_{128} \parallel K_{128}$.

Here, our strategy is to recover K_{128} by independently processing F_0 and F_1^{-1} to find the following match.

$$F_0(S^{-10} \oplus (0, K_{128} \| K_{128}, 0, 0, 0)) \stackrel{?}{=} F_1^{-1}(S^0 \oplus (0, K_{128} \| K_{128}, 0, 0, 0)).$$

To evaluate the attack complexity, we consider the following parameters.

- G_0: a set of bits of K_{128} that are guessed for computing F_0.
- G_1: a set of bits of K_{128} that are guessed for computing F_1^{-1}.
- G_2: a set of bits in the intersection of G_0 and G_1.
- x bits can match after processing F_0 and F_1^{-1}.

Suppose that the union of G_0 and G_1 covers all the bits of K_{128}. The attack exhaustively guesses G_2 and performs the following procedure for each guess.

1. F_0 is computed $2^{|G_0|-|G_2|}$ times and the results are stored in a table T. (Because $|G_1|-|G_2|$ bits are unknown, only a part of the state is computed.)
2. F_1^{-1} is computed $2^{|G_1|-|G_2|}$ times and for each result we check the match with any entry in T.
3. There are $2^{|G_0|-|G_2|+|G_1|-|G_2|}$ combinations, and the number of valid matches reduces to $2^{|G_0|-|G_2|+|G_1|-|G_2|-x}$ after matching the x bits.
4. Check the correctness of the guess by using one plaintext-ciphertext pair.

In the end, F_0 is computed $2^{|G_2|} \cdot 2^{|G_0|-|G_2|} = 2^{|G_0|}$ times. Similarly, F_1^{-1} is computed $2^{|G_1|}$ times. The number of the total candidates after the x-bit match is $2^{|G_2|} \cdot 2^{|G_0|-|G_2|+|G_1|-|G_2|-x} = 2^{|G_0|+|G_1|-|G_2|-x}$. Hence, the key K_{128} is recovered with complexity

$$\max(2^{|G_0|}, 2^{|G_1|}, 2^{|G_0|+|G_1|-|G_2|-x}).$$

Suppose that we choose $|G_0|$ and $|G_1|$ to be balanced i.e., $|G_0| = |G_1|$. Then, the complexity is

$$\max(2^{|G_0|}, 2^{2|G_0|-|G_2|-x}).$$

Two terms are balanced when $x = |G_0| - |G_2|$. Hence, the number of matched bits in the middle of two functions must be greater than or equal to the number of independently guessed bits to compute F_0 and F_1^{-1}.

In the attack below, we choose $|G_0| = |G_1| = 127$ and $|G_2| = 126$ (equivalently $|G_2| - |G_0| = |G_2| - |G_1| = 1$) in order to aim $x = 1$-bit match in the middle, which maximizes the number of attacked rounds.

Full Diffusion Rounds. We found that `StepUpdate` was designed to have good diffusion in the forward direction. Thus, once the state is recovered, the attacker can perform the partial computation in the backward direction longer than the forward direction. We set G_0 and G_1 as follows.

$G_0 = \{1, 2, \cdots, 127\}$	Bit position 0 is unknown.
$G_1 = \{0, 1, \cdots, 7, 9, 10, \cdots, 127\}$	Bit position 8 is unknown.

Those will lead to 4 matching bits after the 4-step forward computation and the 6-step backward computation. The analysis of the diffusion is given in Table 5. In the end, K_{128} can be recovered faster than the exhaustive search by 1 bit, i.e., with complexity 2^{127}.

Remarks. The matching state does not have to be a border of a step. It can be defined on a border of a round, or even in some more complicated way. We did not find the extension of the number of attacked steps even with this way.

As can be seen in Table 5, the updated register in step i is independent of the update function in step $i + 1$ in the forward direction, and starts to impact from step $i + 2$. By modifying this point, the diffusion speed can increase faster, which makes this attack harder.

8 Conclusion

This work provides a comprehensive analysis of the components of MORUS. In particular, we show that MORUS-1280's keystream exhibits a correlation of 2^{-76} between certain ciphertext bits. This enables a plaintext recovery attack in the broadcast setting, using about 2^{152} blocks of data. While the amount of data required is impractical, this seems to violate the security claims of MORUS-1280 because the attack works even if the key is refreshed regularly. Moreover, the broadcast setting is practically relevant, as was shown with attacks against RC4 as used in TLS [1].

We have shared an earlier version of this paper with the authors of MORUS and they agree with the technical details of the keystream bias. However they consider that it is not a significant weakness in practice because it requires more

than 2^{64} ciphertexts bits. In the context of the CAESAR competition, we believe that certificational attacks such as this one should be taken into account, in order to select a portfolio of candidates that reflects the state of the art in terms of cryptographic design.

Acknowledgments. The results presented here were originally found during the Flexible Symmetric Cryptography workshop held at the Lorentz Center in Leiden, Netherlands. The authors would like to thank Meltem Sonmez Turan, who participated in the initial discussion. The second author was supported by the European Union's H2020 grant 644052 (HECTOR). The fourth and sixth authors are partially supported by the French Agence Nationale de la Recherche through the BRUTUS project under Contract ANR-14-CE28-0015. The fifth author was supported by EPSRC Grant EP/M013472/1.

References

1. AlFardan, N.J., Bernstein, D.J., Paterson, K.G., Poettering, B., Schuldt, J.C.N.: On the security of RC4 in TLS. In: USENIX Security Symposium 2013. pp. 305–320. USENIX Association (2013)
2. Ashur, T., Eichlseder, M., Lauridsen, M.M., Leurent, G., Minaud, B., Rotella, Y., Sasaki, Y., Viguier, B.: Cryptanalysis of MORUS. Cryptology ePrint Archive, Report 2018/464 (2018), https://eprint.iacr.org/2018/464
3. Ashur, T., Rijmen, V.: On linear hulls and trails. In: Dunkelman, O., Sanadhya, S.K. (eds.) Progress in Cryptology – INDOCRYPT 2016. LNCS, vol. 10095, pp. 269–286 (2016)
4. CAESAR Committee: CAESAR: Competition for authenticated encryption: Security, applicability, and robustness. Call for submissions (2013), http://competitions.cr.yp.to/caesar-call.html
5. Duong, T., Rizzo, J.: Here come the ⊕ ninjas. Ekoparty (2011)
6. Dwivedi, A.D., Klouček, M., Morawiecki, P., Nikolić, I., Pieprzyk, J., Wójtowicz, S.: SAT-based cryptanalysis of authenticated ciphers from the CAESAR competition. Cryptology ePrint Archive, Report 2016/1053 (2016), https://eprint.iacr.org/2016/1053
7. Dwivedi, A.D., Morawiecki, P., Wójtowicz, S.: Differential and rotational cryptanalysis of round-reduced MORUS. In: Samarati, P., Obaidat, M.S., Cabello, E. (eds.) E-Business and Telecommunications – ICETE/SECRYPT 2017. pp. 275–284. SciTePress (2017)
8. Dworkin, M.J.: NIST SP 800-38D: Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. National Institute of Standards and Technology (NIST) Special Publication (SP) (2007), https://www.nist.gov/node/562956
9. Kales, D., Eichlseder, M., Mendel, F.: Note on the robustness of CAESAR candidates. IACR Cryptology ePrint Archive, Report 2017/1137 (2017), https://eprint.iacr.org/2017/1137
10. Mantin, I., Shamir, A.: A practical attack on broadcast RC4. In: Fast Software Encryption – FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer (2001)

11. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology – EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer (1993)
12. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) Advances in Cryptology – EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer (1992)
13. McGrew, D.A., Viega, J.: The security and performance of the Galois/Counter Mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) Progress in Cryptology – INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer (2004), see also: https://eprint.iacr.org/2004/193
14. Mileva, A., Dimitrova, V., Velichkov, V.: Analysis of the authenticated cipher MORUS (v1). In: Pasalic, E., Knudsen, L.R. (eds.) Cryptography and Information Security in the Balkans – BalkanCryptSec 2015. LNCS, vol. 9540, pp. 45–59. Springer (2015)
15. Minaud, B.: Linear biases in AEGIS keystream. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography – SAC 2014. LNCS, vol. 8781, pp. 290–305. Springer (2014), see also: https://eprint.iacr.org/2018/292
16. Salam, M.I., Simpson, L., Bartlett, H., Dawson, E., Pieprzyk, J., Wong, K.K.: Investigating cube attacks on the authenticated encryption stream cipher MORUS. In: IEEE Trustcom/BigDataSE/ICESS 2017. pp. 961–966. IEEE (2017)
17. Shi, T., Guan, J., Li, J., Zhang, P.: Improved collision cryptanalysis of authenticated cipher morus. In: Artificial Intelligence and Industrial Engineering – AIIE 2016. Advances in Intelligent Systems Research, vol. 133, pp. 429–432. Atlantis Press (2016)
18. Vaudenay, S., Vizár, D.: Under pressure: Security of CAESAR candidates beyond their guarantees. Cryptology ePrint Archive, Report 2017/1147 (2017), https://eprint.iacr.org/2017/1147
19. Wu, H., Huang, T.: The authenticated cipher MORUS (v2). Submission to CAESAR: Competition for Authenticated Encryption. Security, Applicability, and Robustness (Round 3 and Finalist) (September 2016), http://competitions.cr.yp.to/round3/morusv2.pdf
20. Wu, H., Preneel, B.: AEGIS: A fast authenticated encryption algorithm. In: Lange, T., Lauter, K.E., Lisonek, P. (eds.) Selected Areas in Cryptography – SAC 2013. LNCS, vol. 8282, pp. 185–201. Springer (2013), see also: https://eprint.iacr.org/2013/695
21. Wu, H., Preneel, B.: AEGIS: A fast authenticated encryption algorithm (v1.1). Submission to CAESAR: Competition for Authenticated Encryption. Security, Applicability, and Robustness (Round 3 and Finalist) (September 2016), http://competitions.cr.yp.to/round3/aegisv11.pdf

A Trail Equations

In this section, we provide the full trail equation for MORUS-1280. Trail equations for MORUS-640 are available in the full version of this paper [2]. In each case, we decompose the right-hand side of the equality (involving state bits) into connected components, and compute the weight of each of these connected components. If we assume that distinct state bits are uniformly random and independent, then each connected component is independent. By the Piling-Up

Lemma, it follows that the weight of the full equation is equal to the sum of the weights of the connected components.

A.1 Trail Equation for full MORUS-1280

$$
\begin{aligned}
& C^0_{51} \oplus C^0_{115} \oplus C^0_{179} \oplus C^0_{243} \oplus C^1_{0} \oplus C^1_{25} \oplus C^1_{33} \oplus C^1_{55} \oplus C^1_{64} \oplus C^1_{89} \\
\oplus\ & C^1_{97} \oplus C^1_{119} \oplus C^1_{128} \oplus C^1_{153} \oplus C^1_{161} \oplus C^1_{183} \oplus C^1_{192} \oplus C^1_{217} \oplus C^1_{225} \oplus C^1_{247} \\
\oplus\ & C^2_{4} \oplus C^2_{7} \oplus C^2_{29} \oplus C^2_{37} \oplus C^2_{38} \oplus C^2_{46} \oplus C^2_{51} \oplus C^2_{68} \oplus C^2_{71} \oplus C^2_{93} \\
\oplus\ & C^2_{101} \oplus C^2_{102} \oplus C^2_{110} \oplus C^2_{115} \oplus C^2_{132} \oplus C^2_{135} \oplus C^2_{157} \oplus C^2_{165} \oplus C^2_{166} \oplus C^2_{174} \\
\oplus\ & C^2_{179} \oplus C^2_{196} \oplus C^2_{199} \oplus C^2_{221} \oplus C^2_{229} \oplus C^2_{230} \oplus C^2_{238} \oplus C^2_{243} \oplus C^3_{11} \oplus C^3_{20} \\
\oplus\ & C^3_{42} \oplus C^3_{50} \oplus C^3_{75} \oplus C^3_{84} \oplus C^3_{106} \oplus C^3_{114} \oplus C^3_{139} \oplus C^3_{148} \oplus C^3_{170} \oplus C^3_{178} \\
\oplus\ & C^3_{203} \oplus C^3_{212} \oplus C^3_{234} \oplus C^3_{242} \oplus C^4_{24} \oplus C^4_{88} \oplus C^4_{152} \oplus C^4_{216} \\
=\ & S^1_{2,0} \cdot S^1_{3,192} \oplus S^1_{2,0} \cdot S^1_{3,0} \oplus S^1_{2,64} \cdot S^1_{3,0} \oplus S^1_{2,64} \cdot S^1_{3,64} \\
& \quad \oplus S^1_{2,128} \cdot S^1_{3,64} \oplus S^1_{2,128} \cdot S^1_{3,128} \oplus S^1_{2,192} \cdot S^1_{3,128} \oplus S^1_{2,192} \cdot S^1_{3,192} && \text{weight } 3 \\
\oplus\ & S^2_{2,4} \cdot S^2_{3,4} \oplus S^2_{2,68} \cdot S^2_{3,4} \oplus S^2_{2,68} \cdot S^2_{3,68} \oplus S^2_{2,132} \cdot S^2_{3,68} \\
& \quad \oplus S^2_{2,132} \cdot S^2_{3,132} \oplus S^2_{2,196} \cdot S^2_{3,132} \oplus S^2_{2,196} \cdot S^2_{3,196} \oplus S^2_{2,4} \cdot S^2_{3,196} && \text{weight } 3 \\
\oplus\ & S^2_{2,102} \cdot S^2_{3,38} \oplus S^2_{2,102} \cdot S^2_{3,102} \oplus S^2_{2,166} \cdot S^2_{3,102} \oplus S^2_{2,166} \cdot S^2_{3,166} \\
& \quad \oplus S^2_{2,230} \cdot S^2_{3,166} \oplus S^2_{2,230} \cdot S^2_{3,230} \oplus S^2_{2,38} \cdot S^2_{3,230} \oplus S^2_{2,38} \cdot S^2_{3,38} && \text{weight } 3 \\
\oplus\ & S^3_{2,42} \cdot S^3_{3,42} \oplus S^3_{2,106} \cdot S^3_{3,42} \oplus S^3_{2,106} \cdot S^3_{3,106} \oplus S^3_{2,170} \cdot S^3_{3,106} \\
& \quad \oplus S^3_{2,170} \cdot S^3_{3,170} \oplus S^3_{2,234} \cdot S^3_{3,170} \oplus S^3_{2,234} \cdot S^3_{3,234} \oplus S^3_{2,42} \cdot S^3_{3,234} && \text{weight } 3 \\
\oplus\ & S^0_{1,51} \cdot S^0_{2,51} \oplus S^0_{1,51} \oplus S^0_{2,51} \cdot S^0_{3,51} \oplus S^0_{3,51} && \text{weight } 1 \\
\oplus\ & S^0_{1,115} \cdot S^0_{2,115} \oplus S^0_{1,115} \oplus S^0_{2,115} \cdot S^0_{3,115} \oplus S^0_{3,115} && \text{weight } 1 \\
\oplus\ & S^0_{1,179} \cdot S^0_{2,179} \oplus S^0_{1,179} \oplus S^0_{2,179} \cdot S^0_{3,179} \oplus S^0_{3,179} && \text{weight } 1 \\
\oplus\ & S^0_{1,243} \cdot S^0_{2,243} \oplus S^0_{1,243} \oplus S^0_{2,243} \cdot S^0_{3,243} \oplus S^0_{3,243} && \text{weight } 1 \\
\oplus\ & S^1_{1,25} \cdot S^1_{2,25} \oplus S^1_{1,25} \oplus S^1_{2,25} \cdot S^1_{3,25} \oplus S^1_{3,25} && \text{weight } 1 \\
\oplus\ & S^1_{1,33} \cdot S^1_{2,33} \oplus S^1_{1,33} \oplus S^1_{2,33} \cdot S^1_{3,33} \oplus S^1_{3,33} && \text{weight } 1 \\
\oplus\ & S^1_{1,55} \cdot S^1_{2,55} \oplus S^1_{1,55} \oplus S^1_{2,55} \cdot S^1_{3,55} \oplus S^1_{3,55} && \text{weight } 1 \\
\oplus\ & S^1_{1,89} \cdot S^1_{2,89} \oplus S^1_{1,89} \oplus S^1_{2,89} \cdot S^1_{3,89} \oplus S^1_{3,89} && \text{weight } 1 \\
\oplus\ & S^1_{1,97} \cdot S^1_{2,97} \oplus S^1_{1,97} \oplus S^1_{2,97} \cdot S^1_{3,97} \oplus S^1_{3,97} && \text{weight } 1 \\
\oplus\ & S^1_{1,119} \cdot S^1_{2,119} \oplus S^1_{1,119} \oplus S^1_{2,119} \cdot S^1_{3,119} \oplus S^1_{3,119} && \text{weight } 1 \\
\oplus\ & S^1_{1,153} \cdot S^1_{2,153} \oplus S^1_{1,153} \oplus S^1_{2,153} \cdot S^1_{3,153} \oplus S^1_{3,153} && \text{weight } 1 \\
\oplus\ & S^1_{1,161} \cdot S^1_{2,161} \oplus S^1_{1,161} \oplus S^1_{2,161} \cdot S^1_{3,161} \oplus S^1_{3,161} && \text{weight } 1 \\
\oplus\ & S^1_{1,183} \cdot S^1_{2,183} \oplus S^1_{1,183} \oplus S^1_{2,183} \cdot S^1_{3,183} \oplus S^1_{3,183} && \text{weight } 1 \\
\oplus\ & S^1_{1,217} \cdot S^1_{2,217} \oplus S^1_{1,217} \oplus S^1_{2,217} \cdot S^1_{3,217} \oplus S^1_{3,217} && \text{weight } 1
\end{aligned}
$$

$\oplus\ S^1_{1,225} \cdot S^1_{2,225} \oplus S^1_{1,225} \oplus S^1_{2,225} \cdot S^1_{3,225} \oplus S^1_{3,225}$ weight 1

$\oplus\ S^1_{1,247} \cdot S^1_{2,247} \oplus S^1_{1,247} \oplus S^1_{2,247} \cdot S^1_{3,247} \oplus S^1_{3,247}$ weight 1

$\oplus\ S^2_{1,7} \cdot S^2_{2,7} \oplus S^2_{1,7} \oplus S^2_{2,7} \cdot S^2_{3,7} \oplus S^2_{3,7}$ weight 1

$\oplus\ S^2_{1,29} \cdot S^2_{2,29} \oplus S^2_{1,29} \oplus S^2_{2,29} \cdot S^2_{3,29} \oplus S^2_{3,29}$ weight 1

$\oplus\ S^2_{1,37} \cdot S^2_{2,37} \oplus S^2_{1,37} \oplus S^2_{2,37} \cdot S^2_{3,37} \oplus S^2_{3,37}$ weight 1

$\oplus\ S^2_{1,51} \cdot S^2_{2,51} \oplus S^2_{1,51} \oplus S^2_{2,51} \cdot S^2_{3,51} \oplus S^2_{3,51}$ weight 1

$\oplus\ S^2_{1,71} \cdot S^2_{2,71} \oplus S^2_{1,71} \oplus S^2_{2,71} \cdot S^2_{3,71} \oplus S^2_{3,71}$ weight 1

$\oplus\ S^2_{1,93} \cdot S^2_{2,93} \oplus S^2_{1,93} \oplus S^2_{2,93} \cdot S^2_{3,93} \oplus S^2_{3,93}$ weight 1

$\oplus\ S^2_{1,101} \cdot S^2_{2,101} \oplus S^2_{1,101} \oplus S^2_{2,101} \cdot S^2_{3,101} \oplus S^2_{3,101}$ weight 1

$\oplus\ S^2_{1,115} \cdot S^2_{2,115} \oplus S^2_{1,115} \oplus S^2_{2,115} \cdot S^2_{3,115} \oplus S^2_{3,115}$ weight 1

$\oplus\ S^2_{1,135} \cdot S^2_{2,135} \oplus S^2_{1,135} \oplus S^2_{2,135} \cdot S^2_{3,135} \oplus S^2_{3,135}$ weight 1

$\oplus\ S^2_{1,157} \cdot S^2_{2,157} \oplus S^2_{1,157} \oplus S^2_{2,157} \cdot S^2_{3,157} \oplus S^2_{3,157}$ weight 1

$\oplus\ S^2_{1,165} \cdot S^2_{2,165} \oplus S^2_{1,165} \oplus S^2_{2,165} \cdot S^2_{3,165} \oplus S^2_{3,165}$ weight 1

$\oplus\ S^2_{1,179} \cdot S^2_{2,179} \oplus S^2_{1,179} \oplus S^2_{2,179} \cdot S^2_{3,179} \oplus S^2_{3,179}$ weight 1

$\oplus\ S^2_{1,199} \cdot S^2_{2,199} \oplus S^2_{1,199} \oplus S^2_{2,199} \cdot S^2_{3,199} \oplus S^2_{3,199}$ weight 1

$\oplus\ S^2_{1,221} \cdot S^2_{2,221} \oplus S^2_{1,221} \oplus S^2_{2,221} \cdot S^2_{3,221} \oplus S^2_{3,221}$ weight 1

$\oplus\ S^2_{1,229} \cdot S^2_{2,229} \oplus S^2_{1,229} \oplus S^2_{2,229} \cdot S^2_{3,229} \oplus S^2_{3,229}$ weight 1

$\oplus\ S^2_{1,243} \cdot S^2_{2,243} \oplus S^2_{1,243} \oplus S^2_{2,243} \cdot S^2_{3,243} \oplus S^2_{3,243}$ weight 1

$\oplus\ S^3_{1,11} \cdot S^3_{2,11} \oplus S^3_{1,11} \oplus S^3_{2,11} \cdot S^3_{3,11} \oplus S^3_{3,11}$ weight 1

$\oplus\ S^3_{1,75} \cdot S^3_{2,75} \oplus S^3_{1,75} \oplus S^3_{2,75} \cdot S^3_{3,75} \oplus S^3_{3,75}$ weight 1

$\oplus\ S^3_{1,139} \cdot S^3_{2,139} \oplus S^3_{1,139} \oplus S^3_{2,139} \cdot S^3_{3,139} \oplus S^3_{3,139}$ weight 1

$\oplus\ S^3_{1,203} \cdot S^3_{2,203} \oplus S^3_{1,203} \oplus S^3_{2,203} \cdot S^3_{3,203} \oplus S^3_{3,203}$ weight 1

$\oplus\ S^2_{0,0} \cdot S^2_{1,0}$ weight 1

$\oplus\ S^2_{0,64} \cdot S^2_{1,64}$ weight 1

$\oplus\ S^2_{0,128} \cdot S^2_{1,128}$ weight 1

$\oplus\ S^2_{0,192} \cdot S^2_{1,192}$ weight 1

$\oplus\ S^3_{0,230} \cdot S^3_{1,230}$ weight 1

$\oplus\ S^2_{2,46} \cdot S^2_{3,46}$ weight 1

$\oplus\ S^2_{2,110} \cdot S^2_{3,110}$ weight 1

$\oplus\ S^2_{2,174} \cdot S^2_{3,174}$ weight 1

$\oplus\ S^2_{2,238} \cdot S^2_{3,238}$ weight 1

$\oplus\ S^2_{3,64} \cdot S^2_{4,0}$ weight 1

$\oplus\ S^2_{3,128} \cdot S^2_{4,64}$ weight 1

$\oplus\ S^2_{3,192} \cdot S^2_{4,128}$	weight 1
$\oplus\ S^2_{3,0} \cdot S^2_{4,192}$	weight 1
$\oplus\ S^3_{0,38} \cdot S^3_{1,38}$	weight 1
$\oplus\ S^3_{0,102} \cdot S^3_{1,102}$	weight 1
$\oplus\ S^3_{0,166} \cdot S^3_{1,166}$	weight 1
$\oplus\ S^3_{2,20} \cdot S^3_{3,20}$	weight 1
$\oplus\ S^3_{2,50} \cdot S^3_{3,50}$	weight 1
$\oplus\ S^3_{2,84} \cdot S^3_{3,84}$	weight 1
$\oplus\ S^3_{2,114} \cdot S^3_{3,114}$	weight 1
$\oplus\ S^3_{2,148} \cdot S^3_{3,148}$	weight 1
$\oplus\ S^3_{2,178} \cdot S^3_{3,178}$	weight 1
$\oplus\ S^3_{2,212} \cdot S^3_{3,212}$	weight 1
$\oplus\ S^3_{2,242} \cdot S^3_{3,242}$	weight 1
$\oplus\ S^4_{2,24} \cdot S^4_{3,24}$	weight 1
$\oplus\ S^4_{2,88} \cdot S^4_{3,88}$	weight 1
$\oplus\ S^4_{2,152} \cdot S^4_{3,152}$	weight 1
$\oplus\ S^4_{2,216} \cdot S^4_{3,216}$	weight 1

The total weight of the trail is 76.

Table 4: Differential propagation through 3 Steps. Five lines for round i denote the difference of $S_0, \cdots, S_4$ after the round i transformation.

Round	State difference				Weight	Accumulated probability
Ini	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	–
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
1	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	1
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
2	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	1
	0000400000000000	0000000000000000	0000000000000000	0000000000000000	1	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
3	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	1
	0000400000000000	0000000000000000	0000000000000000	0000000000000000	1	
	0000004000000000	0000000000000000	0000000000000000	0000000000000000	1	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
4	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	1
	0000000000000000	0000000000000000	0000400000000000	0000000000000000	1	
	0000004000000000	0000000000000000	0000000000000000	0000000000000000	1	
	0020000000000080	0000000000000000	0000000000000000	0000000000000000	2	
	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	
5	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	2^{-1}
	0000000000000000	0000000000000000	0000400000000000	0000000000000000	1	
	0000000000000000	0000000000000000	0000000000000000	0000004000000000	1	
	0020000000000080	0000000000000000	0000000000000000	0000000000000000	2	
	0000040000000010	0000000000000000	0000000000000000	0000000000000000	2	
6	0000000000100004	0000000000000000	0000000000000000	0000000000000000	2	2^{-3}
	0000000000000000	0000000000000000	0000400000000000	0000000000000000	1	
	0000000000000000	0000000000000000	0000000000000000	0000004000000000	1	
	0000000000000000	0000000000000000	0000000000000000	0020000000000080	2	
	0000040000000010	0000000000000000	0000000000000000	0000000000000000	2	
7	0000000000100004	0000000000000000	0000000000000000	0000000000000000	2	2^{-6}
	0004400001000000	0000000000000000	0000000010000000	0000000000000000	4	
	0000000000000000	0000000000000000	0000000000000000	0000004000000000	1	
	0000000000000000	0000000000000000	0000000000000000	0020000000000080	2	
	0000000000000000	0000000000000000	0000040000000010	0000000000000000	2	
8	0000000000000000	0000000000100004	0000000000000000	0000000000000000	2	2^{-10}
	0004400001000000	0000000000000000	0000000010000000	0000000000000000	4	
	0400014000000000	0000000000000000	0000000000000000	0000000000001000	4	
	0000000000000000	0000000000000000	0000000000000000	0020000000000080	2	
	0000000000000000	0000000000000000	0000040000000010	0000000000000000	2	
9	0000000000000000	0000000000100004	0000000000000000	0000000000000000	2	2^{-14}
	0000000010000000	0000000000000000	0004400001000000	0000000000000000	4	
	0400014000000000	0000000000000000	0000000000000000	0000000000001000	4	
	0220000080000080	0000000000000000	0000000800000000	1000000000004000	7	
	0000000000000000	0000000000000000	0000040000000010	0000000000000000	2	
10	0000000000000000	0000000000100004	0000000000000000	0000000000000000	2	2^{-20}
	0000000010000000	0000000000000000	0004400001000000	0000000000000000	4	
	0000000000000000	0000000000000000	0000000000001000	0400014000000000	4	
	0220000080000080	0000000000000000	0000000800000000	1000000000004000	7	
	4000140000000010	0000000000000000	0000400000000100	0000000000010000	7	
11	0000100000100044	0000000200008000	0001000000000000	0000000008000200	9	2^{-28}
	0000000010000000	0000000000000000	0004400001000000	0000000000000000	4	
	0000000000000000	0000000000000000	0000000000001000	0400014000000000	4	
	0000000000000000	0000000800000000	1000000000004000	0220000080000080	7	
	4000140000000010	0000000000000000	0000400000000100	0000000000010000	7	
12	0000100000100044	0000000200008000	0001000000000000	0000000008000200	9	2^{-39}
	0004500005000400	0000000000000000	0040000100000040	4000000000000000	10	
	0000000000000000	0000000000000000	0000000000001000	0400014000000000	4	
	0000000000000000	0000000800000000	1000000000004000	0220000080000080	7	
	0000400000000100	0000000000010000	4000140000000010	0000000000000000	7	
13	0000000008000200	0000100000100044	0000000200008000	0001000000000000	9	2^{-53}
	0004500005000400	0000000000000000	0040000100000040	4000000000000000	10	
	0400114000040000	0020000000000080	0004000000400000	0000800100005002	14	
	0000000000000000	0000000800000000	1000000000004000	0220000080000080	7	
	0000400000000100	0000000000010000	4000140000000010	0000000000000000	7	
14	0000000008000200	0000100000100044	0000000200008000	0001000000000000	9	2^{-69}
	0040000100000040	4000000000000000	0004500005000400	0000000000000000	10	
	0400114000040000	0020000000000080	0004000000400000	0000800100005002	14	
	0228000280020080	0000040000000000	2000008000202008	1000004000004021	18	
	0000400000000100	0000000000010000	4000140000000010	0000000000000000	7	
15	0000000008000200	0000100000100044	0000000200008000	0001000000000000	9	–
	0040000100000040	4000000000000000	0004500005000400	0000000000000000	10	
	0020000000000080	0004000000400000	0000800100005002	0400114000040000	14	
	0228000280020080	0000040000000000	2000008000202008	1000004000004021	18	
	0000400000000100	0000000000010000	4000140000000010	0000000000000000	7	
ΔT			600080830020f00a	1405414005044421		2^{-88}

Table 5: Analysis of the diffusion and matching bits over 10 steps. '0' and '1' denote that the state bit can and cannot be computed from a partial knowledge of K_{128}, respectively. After the partial computations from each direction, 4 bits of S^{-6} can match.

Round	State Difference			
$S^{-10} \oplus K_{128}$	0000000000000000	0000000000000000	0000000000000000	0000000000000000
	0000000000000000	0000000000000001	0000000000000000	0000000000000001
	0000000000000000	0000000000000000	0000000000000000	0000000000000000
	0000000000000000	0000000000000000	0000000000000000	0000000000000000
	0000000000000000	0000000000000000	0000000000000000	0000000000000000
1	0000000000002000	0000000000000000	0000000000002000	0000000000000000
	0000000000000000	0000400000000000	0000000000000000	0000400000000000
	0008000000000000	0000000000000000	0008000000000000	0000000000000000
	0000000000100000	0020000000000000	0000000000100000	0020000000000000
	0000000000020000	0084000000000000	0000000000020000	0084000000000000
2	0800000000000004	0000000204000001	0800000000000004	0000000204000001
	8000000a00000000	0000002110000004	8000000a00000000	0000002110000004
	0400010221000000	008000400a000081	0400010221000000	008000400a000081
	1000050001000244	4200118a08000280	1000050001000244	4200118a08000280
	880004a0a0200858	4840123350000050	880004a0a0200858	4840123350000050
3	023d63c00050a850	00a1442000489380	023d63c00050a850	00a1442000489380
	02b63380056aaa48	00b5563005dcd6c0	02b63380056aaa48	00b5563005dcd6c0
	d42ab556bf5dfcd6	5a26f633a8556aaa	d42ab556bf5dfcd6	5a26f633a8556aaa
	5fbbf556bd556c65	7aab99aaee6bea2c	5fbbf556bd556c65	7aab99aaee6bea2c
	abff7f3ad7feafad	cfff777ffddffd6d	abff7f3ad7feafad	cfff777ffddffd6d
4	fff77dffffdcf57	fefad7efffdffbf7	fff77dffffdcf57	fefad7efffdffbf7
	ffffffffffffbfff	fffbf7fffddfff77	ffffffffffffbfff	fffbf7fffddfff77
	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff
	ffffffffffffffff	fffbffffefffffff	ffffffffffffffff	fffbffffefffffff
	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff
		2-bits match		2-bits match
5	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff
	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff
	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff	ffffffffffffffff
	fffff7ffedffffff	ffffffffedffffff	fffff7ffedffffff	ffffffffedffffff
	ffffffffedffffff	fffff7ffedffffff	ffffffffedffffff	fffff7ffedffffff
6	ffffffffedffffff	fffff7ffedffffff	ffffffffedffffff	fffff7ffedffffff
	fffbf5e7cdfffbf7	fffff7bfcdfff757	fffbf5e7cdfffbf7	fffff7bfcdfff757
	fffbf5e7cdfffbf7	fffff7bfcdfff757	fffbf5e7cdfffbf7	fffff7bfcdfff757
	7ffd75b6cdfff357	fffbf5a6ccfcfb73	7ffd75b6cdfff357	fffbf5a6ccfcfb73
	7ffbf5a6ccfcf373	7ff975b6ccfff353	7ffbf5a6ccfcf373	7ff975b6ccfff353
7	7efbf5a6cc7cf353	7fd975a6cceff353	7efbf5a6cc7cf353	7fd975a6cceff353
	7eb950a4cc78e353	7dd07184cced7153	7eb950a4cc78e353	7dd07184cced7153
	7eb950a4cc78e353	7dd07184ccec7153	7eb950a4cc78e353	7dd07184ccec7153
	7cd051044c6c3153	3e985024cc48a313	7cd051044c6c3153	3e985024cc48a313
	3c905004cc482313	7c9051044c6c2113	3c905004cc482313	7c9051044c6c2113
8	2c905004c4482113	7c9050040c682113	2c905004c4482113	7c9050040c682113
	2810100444082112	5c1010040c402113	2810100444082112	5c1010040c402113
	2810100444082112	1c1010040c402113	2810100444082112	1c1010040c402113
	0c0010040400113	2800000404082112	0c0010040400113	2800000404082112
	0800000404002112	0800100404400113	0800000404002112	0800100404400113
9	0800000404002112	0800100004000112	0800000404002112	0800100004000112
	0000000404000102	0000100004000110	0000000404000102	0000100004000110
	0000000404000102	0000000004000110	0000000404000102	0000000004000110
	0000000004000110	0000000000000102	0000000004000110	0000000000000102
	0000000000000100	0000000004000110	0000000000000100	0000000004000110
10	0000000000000100	0000000004000100	0000000000000100	0000000004000100
	0000000000000000	0000000004000100	0000000000000000	0000000004000100
	0000000000000000	0000000000000100	0000000000000000	0000000000000100
	0000000000000100	0000000000000000	0000000000000100	0000000000000000
	0000000000000000	0000000000000100	0000000000000000	0000000000000100
$S^{0} \oplus K_{128}$	0000000000000000	0000000000000000	0000000000000000	0000000000000000
	0000000000000000	0000000000000100	0000000000000000	0000000000000100
	0000000000000000	0000000000000000	0000000000000000	0000000000000000
	0000000000000000	0000000000000000	0000000000000000	0000000000000000
	0000000000000000	0000000000000000	0000000000000000	0000000000000000