Analysis of the Authenticated Cipher MORUS (v1)

Aleksandra Mileva[1], Vesna Dimitrova[2], and Vesselin Velichkov[3]

[1]Faculty of Computer Science, University "Goce Delčev",
Štip, Republic of Macedonia
aleksandra.mileva@ugd.edu.mk
[2]Faculty of Computer Science and Engineering, University "Ss Cyril and Methodius",
Skopje, Republic of Macedonia
vesna.dimitrova@finki.edu.mk
[3]Laboratory of Algorithmics, Cryptology and Security (LACS),
Université du Luxembourg, SnT / FSTC,
Luxembourg
vesselin.velichkov@uni.lu

Abstract. We present several new observations on the CAESAR candidate MORUS (v1). First, we report a collision on its StateUpdate(S, M) function. Second, we describe a distinguisher in a nonce-reuse scenario with probability 1. Finally, we observe that the differences in some words of the state after the initialization have probabilities significantly higher than the random case. We note that the presented results do not threaten the security of the scheme. This is the first external analysis of the authenticated cipher MORUS.

Keywords: symmetric-key, cryptanalysis, authenticated encryption, CAESAR, MORUS

1 Introduction

In 2013 the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) was announced [2]. It is initiated by the University of Illinois at Chicago, USA and is supported by the US National Institute of Standards and Technology (NIST). CAESAR is similar in nature to widely successful previous competitions such as AES [3] and SHA-3 [4]. While the latter were about standardizing dedicated algorithms for confidentiality [5] and integrity [6] respectively, the goal of CAESAR is to select algorithm/s that can ensure both confidentiality and integrity within a single primitive. It is very likely that these algorithms will end up in standardization and will be implemented and used by the industry worldwide. In total 56 candidates have been submitted to CAESAR and as of March 15, 2014, the competition has entered its public evaluation phase.

In this paper we present the first public analysis of one of the algorithms submitted to CAESAR – the authenticated cipher MORUS [1]. The latter is a very promising design – both efficient and secure – on which no weaknesses have been reported so far. We describe several new observations on MORUS as summarized below:

1. Distinguisher with probability 1 in a nonce-reuse scenario.
2. Differential biases in some words of the state after the initialization.
3. Collision on the StateUpdate(S, M) function of MORUS.

We note that the presented results do not threaten the security of MORUS.

The rest of the paper is organized as follows. We begin with a brief description of MORUS in Sect. 2. A distinguisher on MORUS is presented in Sect. 3, followed by a description of differential

biases in some words of the state after the initialization (Sect. 4). In Sect. 5 we describe a collision on the StateUpdate function and we briefly comment on the possibility of using it for a tag forgery attack (Sect. 6). Section 7 concludes the paper. Theorem proofs and equation derivations are provided in Appendix A and Appendix B. Notation is given in Table 1.

Table 1. Notation.

Symbol	Meaning
$\oplus$	Bit-wise exclusive OR
$\wedge$	Bit-wise AND
$\|\|$	Concatenation
$\lll$	Bit rotation to the left
$\ggg$	Bit rotation to the right
$b^{(n)}$	A sequence of n binary digits $b \in \{0,1\}$
$\|X\|$	Length of the bit string X (in bits)
$\overline{x}$	Negation of all bits of x i.e. $\overline{x} = x \oplus 1^{(n)}$
Rotl_xxx_yy(x,b)	Divide the xxx-bit block x into 4 yy-bit words and rotate each word left by b bits. Example: Rotl_128_32(x,b) is used in MORUS-640 and Rotl_256_64(x,b) is used in MORUS-1280.
Rotr_xxx_yy(x,b)	Analogous to Rotl_xxx_yy(x,b) with a right rotation
LSB, MSB	Least Significant Bit, Most Significant Bit
IV	Initialization Vector (Nonce)

2 Description of MORUS

MORUS is a very efficient family of authenticated encryption schemes using only bitwise operations: bit shift, AND and XOR. The size of the internal state is 640 or 1280 bits resp. for MORUS-640 and MORUS-1280 and is represented as five 128 or 256 bit registers respectively. MORUS supports 128 and 256 bit keys which are loaded into the input state together with a public 128 bit IV and three specified constants.

The main building block of MORUS is the state update function StateUpdate(S, M), where S is the state, and M is a message block with length $|S|/5$. This function consists of 5 rounds with similar operations that update the state S. In each round, only two state elements are modified: one with left rotation with coefficients w_i, and other with Rotl_xxx_yy operation with coefficients b_i, where $i \in \{0,1,2,3,4\}$. The StateUpdate function is shown in Fig. 1.

MORUS operates in four phases: (1) Initialization, (2) Processing of associated data, (3) Encryption and (4) Finalization. In the initialization phase of MORUS-640, five state elements are initialized with an initialization vector IV, a key K, a 128 bit string of ones $1^{(128)}$, and two constants const_0 and const_1. Next the state is updated by 16 applications of the round function StateUpdate and the result is XOR-ed with the key K.

In the second phase the associated data AD is processed using again the StateUpdate function. In the third phase the plaintext P is encrypted in blocks of 128 bits. In the final phase the authentication tag is generated by 8 applications of StateUpdate using the length of associated data

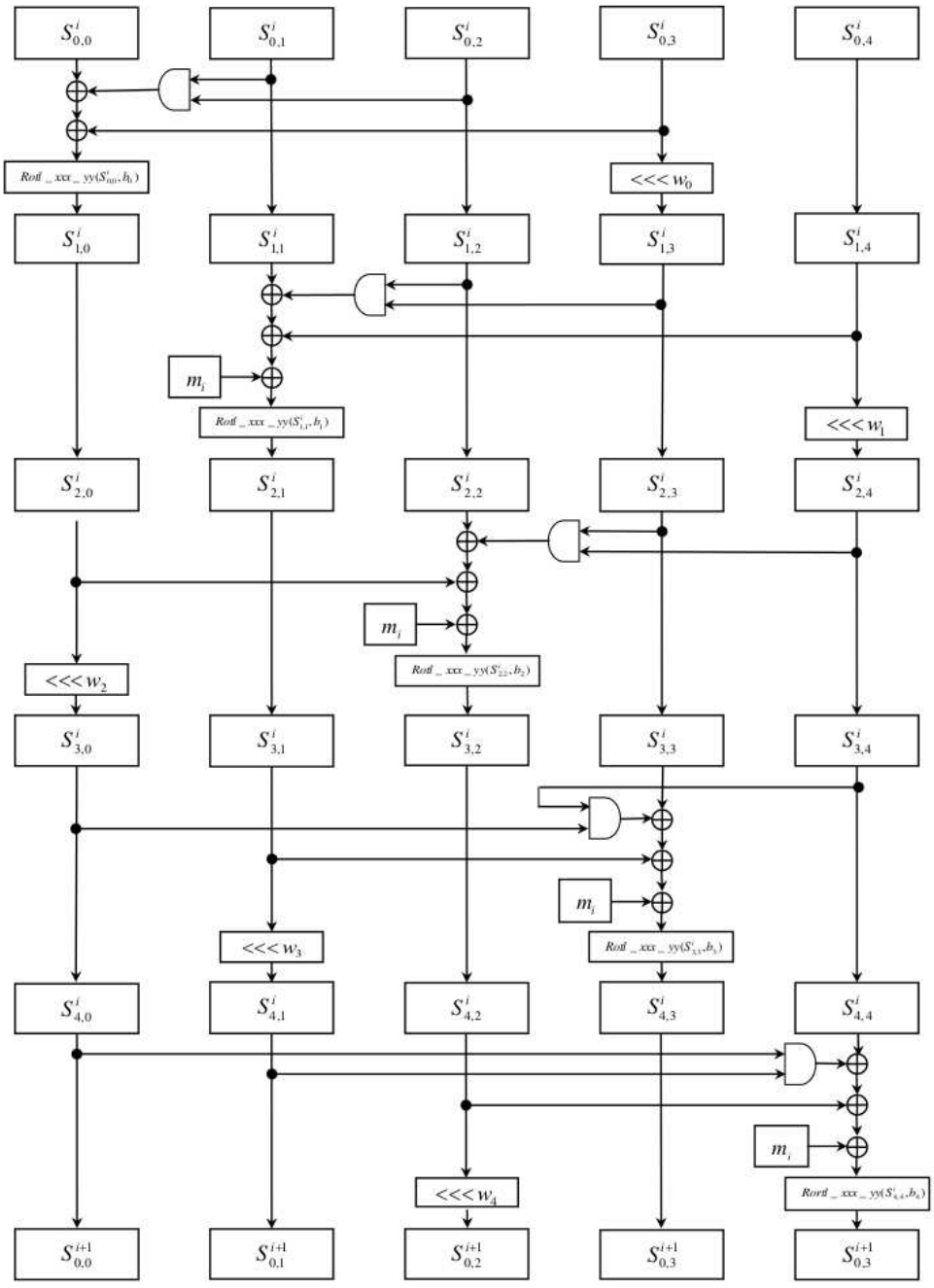

Fig. 1. The StateUpdate(S, M) function of MORUS.

`adlen` and the length of the message `msglen` as additional inputs. The output of the full process in encryption mode is a ciphertext together with an authentication tag of size at most 128 bits.

For the detailed specification of MORUS we refer the reader to the original proposal [1].

3 Distinguisher

In this section we describe a distinguisher on MORUS-640 in a nonce-reuse scenario. An analogous distinguisher also exists for MORUS-1280. We begin with a more formal description of the four phases of MORUS: initialization, processing of associated data, encryption and finalization.

Let $S^0 = (s_0, s_1, s_2, s_3, s_4)$ be the output of the initialization phase. Let the size of the associate data (AD) be 128 bits: $|\mathrm{AD}| = 128$ and let the plaintext P consist of a single 128-bit block M. Then the output after the second phase is expressed as:

$$S^1 = (x_0, x_1, x_2, x_3, x_4) = \mathrm{StateUpdate}(S^0, \mathrm{AD}) \ , \tag{1}$$

and the output of the third phase is the ciphertext C and the state S^2:

$$C = M \oplus x_0 \oplus (x_1 \lll 96) \oplus (x_2 \wedge x_3) \ , \tag{2}$$

$$S^2 = (z_0, z_1, z_2, z_3, z_4) = \mathrm{StateUpdate}(S^1, M) \ . \tag{3}$$

In the final (fourth) phase the authentication tag T is obtained through the following steps:

1. $\mathrm{tmp} = z_3 \oplus (\mathrm{adlen} \ || \ \mathrm{msglen})$
2. $z_4 = z_4 \oplus z_0$
3. For $i = 2$ to 9 do $S^{i+1} = \mathrm{StateUpdate}(S^i, \mathrm{tmp})$
4. $T = \oplus_{i=1}^{4} \ S_i^{10}$

The following theorem is instrumental in the construction of the distinguisher:

Theorem 1. *Let $S^0 = (s_0, s_1, s_2, s_3, s_4)$ be the state of MORUS-640 after initialization under 128 bit secret key K and 128 bit public IV. Let $\mathrm{AD}_1 = 0^{(128)}$ and $\mathrm{AD}_2 = 0^{(127)}||1$ be two 128 bit blocks of authenticated data that differ only in their least significant bit. Finally, let X and Y be the internal states of MORUS after the second phase (processing of associated data) under AD_1 and AD_2 respectively:*

$$X = (x_0, x_1, x_2, x_3, x_4) = \mathrm{StateUpdate}(S^0, \mathrm{AD}_1) \ , \tag{4}$$

$$Y = (y_0, y_1, y_2, y_3, y_4) = \mathrm{StateUpdate}(S^0, \mathrm{AD}_2) \ . \tag{5}$$

Then the following statements are true:

1. $x_0 = y_0$.
2. x_1 *and* y_1 *differ only in the 33-th bit.*
3. x_2 *and* y_2 *differ only in the 89-th bit.*
4. x_3 *and* y_3 *differ only in the 106-th and 107-th bit.*
5. x_4 *and* y_4 *differ only in the 108-th and 115-th bit (with probability* 1*) and in the 33-th bit with probability* 1/2*.*

Note: all bits within the 128 *bit words X and Y are counted starting from MSB (bit* 1*) down to LSB (bit* 128*).*

Proof. Appendix A.

Next we describe the construction of the distinguisher. Let AD_1, AD_2, $X = (x_0, x_1, x_2, x_3, x_4)$ and $Y = (y_0, y_1, y_2, y_3, y_4)$ be as in Theorem 1 and denote $x_j = (x_{j0}, x_{j1}, x_{j2}, x_{j3})$ and $y_j = (y_{j0}, y_{j1}, y_{j2}, y_{j3})$, where $|x_{ji}| = |y_{ji}| = 32$ bits and $0 \leq j \leq 4$, $1 \leq i \leq 3$. Further let M be a 128 bit message block. Then, according to the encryption function of MORUS (see eq. (2)) the two ciphertexts C_1 and C_2 resp. under AD_1 and AD_2 are expressed as:

$$C_1 = M \oplus x_0 \oplus (x_1 \lll 96) \oplus (x_2 \wedge x_3) \ , \tag{6}$$

$$C_2 = M \oplus y_0 \oplus (y_1 \lll 96) \oplus (y_2 \wedge y_3) \ . \tag{7}$$

Note that

$$(x_1 \lll 96) = ((x_{10}, x_{11}, x_{12}, x_{13}) \lll 96) = (x_{13}, x_{10}, x_{11}, x_{12}) \ . \tag{8}$$

Due to Statement 2 of Theorem 1 (see also eq. (44) in Appendix A) it holds that:

$$y_1 = (y_{10}, y_{11}, y_{12}, y_{13}) = (x_{10}, x_{11} \oplus (1||0^{(31)}), x_{12}, x_{13}) \ . \tag{9}$$

Therefore

$$(y_1 \lll 96) = ((x_{10}, x_{11} \oplus (1||0^{(31)}), x_{12}, x_{13}) \lll 96) = (x_{13}, x_{10}, x_{11} \oplus (1||0^{(31)}), x_{12}) \ . \tag{10}$$

It follows that $(x_1 \lll 96)$ and $(y_1 \lll 96)$ differ only in the 65-th bit. Further, in (6) note that

$$(x_2 \wedge x_3) = (x_{20} \wedge x_{30}, x_{21} \wedge x_{31}, x_{22} \wedge x_{32}, x_{23} \wedge x_{33}) \ . \tag{11}$$

Due to Statement 3 of Theorem 1 (see also eq. (57) in Appendix A) it holds that:

$$y_2 = (y_{20}, y_{21}, y_{22}, y_{23}) = (x_{20}, x_{21}, x_{22} \oplus (0^{(24)}||1||0^{(7)}), x_{23}) \ , \tag{12}$$

and due to Statement 4 of Theorem 1 (see also eq. (67) in Appendix A) it holds that:

$$y_3 = (y_{30}, y_{31}, y_{32}, y_{33}) = (x_{30}, x_{31}, x_{32}, x_{33} \oplus (0^{(9)}||11||0^{(21)})) \ . \tag{13}$$

From the last two equations it follows that

$$(y_2 \wedge y_3) = ((x_{20}, x_{21}, x_{22} \oplus (0^{(24)}||1||0^{(7)}), x_{23}) \wedge (x_{30}, x_{31}, x_{32}, x_{33} \oplus (0^{(9)}||11||0^{(21)}))) \tag{14}$$

$$= (x_{20} \wedge x_{30}, x_{21} \wedge x_{31}, (x_{22} \oplus (0^{(24)}||1||0^{(7)})) \wedge x_{32}, x_{23} \wedge (x_{33} \oplus (0^{(9)}||11||0^{(21)}))) \ . \tag{15}$$

From eq. (11) and eq. (15) it follows that $(x_2 \wedge x_3)$ and $(y_2 \wedge y_3)$ differ at most in the 89-th, 106-th and 107-th bit (counting from MSB to LSB), each with probability 1/2.

From the above analysis it follows that C_1 and C_2 differ in 4 bits in total. Namely, they differ in the 65-th bit with probability 1 and in the 89-th, 106-th and 107-th bit with probability 1/2 (for each of the three bits). Therefore given the ciphertext C_1 under message (AD_1, M), an attacker can predict 125 bits of an (unknown) ciphertext C_2 under a different message (AD_2, M) with probability 1. The same probability for a random oracle is 2^{-125} and can therefore clearly be used as a distinguisher.

The same technique can also trivially be extended to distinguish pairs of plaintexts (AD_1, M_1) and (AD_2, M_2) that contain also differences in the message words i.e. $AD_1 \neq AD_2$ and $M_1 \neq M_2$

and such that the messages M_1 and M_2 have the same length as a single block: $|M_1| = |M_2| = 128$. In this case a difference in any k bits of M_1 and M_2 (other than bits 89, 106 and 107) results in a difference in the corresponding k bits of the ciphertexts C_1 and C_2 with probability 1.

In conclusion, we would like to stress that the described distinguisher relies on the re-use of the same IV under the same message M. We note that this scenario is explicitly forbidden in the design document, where it is stated: *In MORUS, each key and IV pair should be used to protect only one message.* [1, Sect. 3]. Therefore the results in this section do not directly harm the security of the algorithm. Nevertheless they remain of high importance in scenarios in which the *IV* is involuntarily re-used e.g. due to an implementation or human error.

4 Differential Biases After Initialization

In this section we describe differential biases in the words of the state after initialization in reduced word versions of MORUS.

We analyze versions of MORUS with 8 bit and 10 bit words (resp. 160 bit and 200 bit state), denoted resp. MORUS-160 and MORUS-200. In the latter, the rotation constants are the same as in the original version modulo the word size. For both small versions we initialize the input state to a fixed random value. Next we try all differences in one word of the IV and we measure the probabilities with which differences in the words of the state after the initialization appear.

For MORUS-160 the best observed probability is $2^{-4.19}$ (see Example 1 below), while for MORUS-200 it is 2^{-6}. These values are higher than what is expected in the random case resp. 2^{-8} and 2^{-10}. Due to the bitwise nature of the cipher, these results also suggest that for the original 32 and 64 bit versions of MORUS resp. MORUS-640 and MORUS-1280 we can expect probabilities significantly higher than resp. 2^{-32} and 2^{-64}.

Example 1 (Differential bias in MORUS-160). Let X be an input state to MORUS-160 initialized to the following random values (in hexadecimal):

$$X = (\texttt{3B}, \texttt{77}, \texttt{E3}, \texttt{DE}, \texttt{F0}, \texttt{EC}, \texttt{D9}, \texttt{DD}, \texttt{2F}, \texttt{D0}, \texttt{F5}, \texttt{C7}, \texttt{DF}, \texttt{7E}, \texttt{C8}, \texttt{DD}, \texttt{E7}, \texttt{1D}, \texttt{3A}, \texttt{73}) \ , \tag{16}$$

where the first (i.e. leftmost) four entries correspond to the IV and the next four entries correspond to the key K i.e. IV = $\texttt{3B77E3DE}$ and $K = \texttt{F0ECD9DD}$. Let $X' = X \oplus \Delta X$ be a second input state that differs from X only in the first (i.e. leftmost) word of the IV by the difference:

$$\Delta X = (\texttt{AB}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \ . \tag{17}$$

Then the sixth word of the output difference ΔY between the corresponding output states Y and Y' after initialization is expected to be $\texttt{4}$ with probability $2^{-4.19}$. The latter can be expressed as the following truncated differential:

$$P(\Delta X = (\texttt{AB}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \xrightarrow{\text{Init.}} \\ \Delta Y = (*, *, *, *, *, \texttt{4}, *, *, *, *, *, *, *, *, *, *, *, *, *, *)) = 2^{-4.19} \ ,$$

where $*$ denotes an unknown difference. □

We note that the above analysis is different than the analysis described by the designers in [1, Sect. 4.1]. There, for a given difference in the IV, the designers estimate the probability of a full state after the initialization as opposed to a state truncated to a single word. Finally, we stress that the described observation does not directly threaten the security of MORUS. Nevertheless its full security implications should further be investigated.

5 Collisions in the StateUpdate(S, M) Function

In this section we present a collision on the internal StateUpdate(S, M) function. We begin by describing an alternative representation of the state update function.

Proposition 1. *Let $M,\ x_i \in \mathbb{Z}_2^n,\ i \geq 0$ and $n \in \{128, 256\}$, and let $w_i \leq n$ and $b_i \leq n/4$ be some rotation constants. Then the following function $F_M : (\mathbb{Z}_2^n)^5 \to (\mathbb{Z}_2^n)^5$ is a permutation on $(\mathbb{Z}_2^n)^5$:*

$$\begin{aligned} &F_M(x_i, x_{i+1}, x_{i+2}, x_{i+3}, x_{i+4}) \\ &= (\text{Rotl\_xxx\_yy}(x_i \oplus (x_{i+1} \wedge x_{i+2}) \oplus x_{i+3} \oplus M, b_i), x_{i+1}, x_{i+2}, (x_{i+3} \lll w_i), x_{i+4}) \ . \end{aligned} \tag{18}$$

Each round of StateUpdate(S, M) can be represented as five applications of the function F_{m_i}, where $m_i = M$ for $i = \{1, 2, 3, 4\}$ and $m_i = 0^{(n)}$ for $i = 0$:

$$(s_i, s_{(i+1)}, s_{(i+2)}, s_{(i+3)}, s_{(i+4)}) = F_{m_i}(s_i, s_{(i+1)}, s_{(i+2)}, s_{(i+3)}, s_{(i+4)}) \ , \tag{19}$$

and the additions $(i+1), \ldots, (i+4)$ in the indices of s are performed modulo 5. From the fact that StateUpdate(S, M) is a composition of permutations on $(\mathbb{Z}_2^n)^5$ it follows that for fixed $M \in \mathbb{Z}_2^n$, the function is a permutation on $(\mathbb{Z}_2^n)^5$. Next, we describe a collision on the F_M function.

Proposition 2. *Let $w_i \leq n$ and $b_i \leq n/4$, where $i \geq 0$ and $n = \{128, 256\}$, be some rotation constants. For all $M_1, M_2 \in \mathbb{Z}_2^n$ and each vector $(x_0, x_1, x_2, x_3, x_4) \in (\mathbb{Z}_2^n)^5$, the following holds:*

$$F_{M_1}(x_0, x_1, x_2, x_3, x_4) = F_{M_2}(M_1 \oplus M_2 \oplus x_0, x_1, x_2, x_3, x_4) \ . \tag{20}$$

Proof.

$$\begin{aligned} &F_{M_2}(M_1 \oplus M_2 \oplus x_0, x_1, x_2, x_3, x_4) \\ &= (\text{Rotl\_xxx\_yy}(M_1 \oplus M_2 \oplus x_0 \oplus (x_1 \wedge x_2) \oplus x_3 \oplus M_2, b_i), x_1, x_2, (x_3 \lll w_i), x_4) && (21) \\ &= (\text{Rotl\_xxx\_yy}(M_1 \oplus x_0 \oplus (x_1 \wedge x_2) \oplus x_3, b_i), x_1, x_2, (x_3 \lll w_i), x_4) && (22) \\ &= F_{M_1}(x_0, x_1, x_2, x_3, x_4) \ . && (23) \end{aligned}$$

□

Proposition 2 is extended to a collision on the state update function due to the following corollary:

Corollary 1. *For all $M_1, M_2 \in \mathbb{Z}_2^n, n = \{128, 256\}$ and each vector $(x_0, x_1, x_2, x_3, x_4) \in (\mathbb{Z}_2^n)^5$, the following holds:*

$$\begin{aligned} &\text{StateUpdate}((x_0, x_1, x_2, x_3, x_4), M_1) = \\ &\text{StateUpdate}((x_0, M_1 \oplus M_2 \oplus x_1, M_1 \oplus M_2 \oplus x_2, M_1 \oplus M_2 \oplus x_3, M_1 \oplus M_2 \oplus x_4), M_2) \ . \end{aligned} \tag{24}$$

In the following section we briefly comment on the possibility of using the collision (24) to construct a tag forgery attack in a nonce-reuse scenario.

6 On Producing a Tag Forgery

Let (M_1, AD_1) be a pair of a 128 bit message block and a 128 bit associated data block encrypted by MORUS under key K and IV. Let $S^0 = (s_0, s_1, s_2, s_3, s_4)$ be the output of the initialization phase and let the output of the next phase (processing of associated data) be $S^1 = (x_0, x_1, x_2, x_3, x_4) = \mathrm{StateUpdate}(S^0, \mathrm{AD}_1)$. We want to find a 128 bit block ΔM and a 128 bit associated data block $\mathrm{AD}_2 \neq \mathrm{AD}_1$ that will relate $\mathrm{StateUpdate}(S^0, \mathrm{AD}_1)$ to $\mathrm{StateUpdate}(S^0, \mathrm{AD}_2)$ as follow:

$$\begin{aligned}
\mathrm{StateUpdate}(S^0, \mathrm{AD}_2) &= (x_0, x_1 \oplus \Delta M, x_2 \oplus \Delta M, x_3 \oplus \Delta M, x_4 \oplus \Delta M) && (25)\\
&= (x_0, x_1, x_2, x_3, x_4) \oplus (0^{(n)}, \Delta M, \Delta M, \Delta M, \Delta M) && (26)\\
&= \mathrm{StateUpdate}(S^0, \mathrm{AD}_1) \oplus (0^{(n)}, \Delta M, \Delta M, \Delta M, \Delta M)\ . && (27)
\end{aligned}$$

If such AD_2 and ΔM exist, then using Corollary 1 (eq. (24)) we can construct the message $M_2 = M_1 \oplus \Delta M$ that will produce a collision in the internal state after the encryption phase:

$$\begin{aligned}
&\mathrm{StateUpdate}(\mathrm{StateUpdate}(S^0, \mathrm{AD}_1), M_1) && (28)\\
&= \mathrm{StateUpdate}((x_0, x_1, x_2, x_3, x_4), M_1) && (29)\\
&= \mathrm{StateUpdate}((x_0, M_1 \oplus M_2 \oplus x_1, M_1 \oplus M_2 \oplus x_2, M_1 \oplus M_2 \oplus x_3, M_1 \oplus M_2 \oplus x_4), M_2) && (30)\\
&= \mathrm{StateUpdate}((x_0, \Delta M \oplus x_1, \Delta M \oplus x_2, \Delta M \oplus x_3, \Delta M \oplus x_4), M_2) && (31)\\
&= \mathrm{StateUpdate}(\mathrm{StateUpdate}(S^0, \mathrm{AD}_2), M_2) && (32)
\end{aligned}$$

Therefore, under the same IV the two messages M_1 and $M_2 = M_1 \oplus \Delta M$ produce the same internal state after the encryption phase. Furthermore, since both pairs (AD_1, M_1) and (AD_2, M_2) have the same `adlen` and `msglen`, the above collision ultimately results in the same tag for the messages M_1 and M_2.

As to finding the required blocks ΔM and AD_2, in Appendix B we show that this is equivalent to solving the following system of equations for $(x_0, x_1, \Delta A, \Delta M)$, where $\Delta A = \mathrm{AD}_1 \oplus \mathrm{AD}_2$:

$$\begin{cases}
\mathrm{Rotr_xxx_yy}(\Delta M \ggg w_3, b_1) = \Delta A\\
\mathrm{Rotr_xxx_yy}(\Delta M \ggg w_4, b_2) = \Delta A\\
\mathrm{Rotr_xxx_yy}(\Delta M, b_3) = (\Delta M \ggg w_3) \oplus \Delta A\\
\mathrm{Rotr_xxx_yy}(\Delta M, b_4) = (x_0 \wedge x_1) \oplus (x_0 \wedge (x_1 \oplus \Delta M)) \oplus (\Delta M \ggg w_4)
\end{cases} \quad (33)$$

We exhaustively searched for a solution to a reduced version of the system (33) composed of the first three equations with 8 bit variables and rotation constants equal to the original ones modulo 8. No solution was found apart from the trivial one: $(\Delta M, \Delta A) = (0, 0)$.

7 Conclusions

In this paper we presented the first external analysis of the authenticated cipher MORUS. In particular we reported the following new observations: (1) distinguisher with probability 1 in a nonce-reuse scenario, (2) differential biases in the words of the state after initialization and (3) collision on the StateUpdate function of MORUS. Our results do not threaten the security of the scheme and indicate that MORUS is a well-designed cipher and a good candidate for the second round of the CAESAR competition.

Acknowledgments. We would like to thank the anonymous reviewers for their time and valuable comments. In particular, we thank Reviewer 2 for pointing out the natural extension of our technique to the case where differences in the message blocks are allowed. Finally, we extend our thanks to the organizers of WG4 Meeting on Authenticated Encryption, COST CryptoAction IC1306, co-located with Eurocrypt 2015, for giving us the opportunity to work on this topic.

References

1. H. Wu, T. Huang, The authenticated cipher MORUS (v1), CAESAR candidate, 15 March 2014.
2. CAESAR - Competition for Authenticated Encryption: Security, Applicability, and Robustness (2014), `http://competitions.cr.yp.to/caesar.html` .
3. National Institute of Standards and Technology, Announcing Request for Candidate Algorithm Nominations for a the Advanced Encryption Standard (AES), Federal Register, vol. 62, pp. 4805148058, September 1997. Available: `http://csrc.nist.gov/archive/aes/pre-round1/aes_9709.htm` .
4. National Institute of Standards and Technology, Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family, Federal Register, vol. 27, pp. 6221262220, November 2007. Available: `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf` .
5. J. Daemen and V. Rijmen, AES and the Wide Trail Design Strategy, in EUROCRYPT (L. R. Knudsen, ed.), vol. 2332 of Lecture Notes in Computer Science, pp. 108109, Springer, 2002.
6. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, Keccak, in EUROCRYPT (T. Johansson and P. Q. Nguyen, eds.), vol. 7881 of Lecture Notes in Computer Science, pp. 313314, Springer, 2013.

A Proof of Theorem 1

Proof. The state update function of MORUS-640 under AD_1 is expressed as follow:

$$(x_0, x_1, x_2, x_3, x_4) = \mathrm{StateUpdate}((s_0, s_1, s_2, s_3, s_4), \mathrm{AD}_1) \ , \tag{34}$$

where

$$x_0 = \mathrm{Rotl\_xxx\_yy}(s_0 \oplus (s_1 \wedge s_2) \oplus s_3, 5) \lll 96 \ , \tag{35}$$
$$x_1 = \mathrm{Rotl\_xxx\_yy}(s_1 \oplus (s_2 \wedge (s_3 \lll 32)) \oplus s_4 \oplus \mathrm{AD}_1, 31) \lll 64 \ , \tag{36}$$
$$x_2 = \mathrm{Rotl\_xxx\_yy}(s_2 \oplus ((s_3 \lll 32) \wedge (s_4 \lll 64)) \oplus (x_0 \ggg 96) \oplus \mathrm{AD}_1, 7) \lll 32 \ , \tag{37}$$
$$x_3 = \mathrm{Rotl\_xxx\_yy}((s_3 \lll 32) \oplus ((s_4 \lll 64) \wedge x_0) \oplus (x_1 \ggg 64) \oplus \mathrm{AD}_1, 22) \ , \tag{38}$$
$$x_4 = \mathrm{Rotl\_xxx\_yy}((s_4 \lll 64) \oplus (x_0 \wedge x_1) \oplus (x_2 \ggg 32) \oplus \mathrm{AD}_1, 13) \ . \tag{39}$$

We recall that each element x_j and y_j of the states $X = (x_0, x_1, x_2, x_3, x_4)$ and $Y = (y_0, y_1, y_2, y_3, y_4)$ is of size 128 bits organized in an array of four 32 bit words. Denote these words as $x_j = (x_{j0}, x_{j1}, x_{j2}, x_{j3})$ and $y_j = (y_{j0}, y_{j1}, y_{j2}, y_{j3})$, where $|x_{ji}| = |y_{ji}| = 32$ bits for $0 \leq j \leq 4$, $1 \leq i \leq 3$.

Proof of Statement 1: $x_0 = y_0$. Since eq. (35) does not depend on the associated data block, it will be the same for both AD_1 and AD_2. It follows that $x_0 = y_0$.

Proof of Statement 2: Difference Between Words x_1 and y_1. In eq. (36) denote

$$a_1 = s_1 \oplus (s_2 \wedge (s_3 \lll 32)) \oplus s_4 \ , \tag{40}$$
$$b_1 = (b_{10}, b_{11}, b_{12}, b_{13}) = \mathrm{Rotl\_xxx\_yy}(a_1 \oplus \mathrm{AD}_1, 31) = \mathrm{Rotl\_xxx\_yy}(a_1, 31) \ . \tag{41}$$

After rotation by 64 (see eq. (36)) we have

$$x_1 = (x_{10}, x_{11}, x_{12}, x_{13}) = (b_{12}, b_{13}, b_{10}, b_{11}) \ . \tag{42}$$

For the second associated data block $\mathrm{AD}_2 = (0^{(127)}||1)$ denote

$$c_1 = (c_{10}, c_{11}, c_{12}, c_{13}) = \mathrm{Rotl\_xxx\_yy}(a_1 \oplus \mathrm{AD}_2, 31) = \mathrm{Rotl\_xxx\_yy}(a_1 \oplus (0^{(127)}||1), 31) \ . \tag{43}$$

Analogously to AD_1, after rotation by 64 (see eq. (36)) we get

$$y_1 = (y_{10}, y_{11}, y_{12}, y_{13}) = (x_{10}, x_{11} \oplus (1||0^{(31)}), x_{12}, x_{13}) \ . \tag{44}$$

For the words of b_1 and c_1, the following equalities hold:

$$c_{10} = b_{10} = x_{12} \ , \tag{45}$$
$$c_{11} = b_{11} = x_{13} \ , \tag{46}$$
$$c_{12} = b_{12} = x_{10} \ , \tag{47}$$
$$c_{13} = b_{13} \oplus (1||0^{(31)}) = x_{11} \oplus (1||0^{(31)}) \ . \tag{48}$$

Therefore x_1 and y_1 differ only in the 33-th bit (counting from MSB to LSB).

Proof of Statement 3: Difference Between Words x_2 and y_2. In eq. (37) denote

$$a_2 = s_2 \oplus ((s_3 \lll 32) \wedge (s_4 \lll 64)) \oplus (x_0 \ggg 96) \ , \tag{49}$$
$$b_2 = (b_{20}, b_{21}, b_{22}, b_{23}) = \mathrm{Rotl\_xxx\_yy}(a_2 \oplus \mathrm{AD}_1, 7) = \mathrm{Rotl\_xxx\_yy}(a_2, 7) \ . \tag{50}$$

After rotation by 32 (see eq. (36)) we have

$$x_2 = (x_{20}, x_{21}, x_{22}, x_{23}) = (b_{21}, b_{22}, b_{23}, b_{20}) \ . \tag{51}$$

For the second associated data block $\mathrm{AD}_2 = (0^{(127)}||1)$ denote

$$c_2 = (c_{20}, c_{21}, c_{22}, c_{23}) = \mathrm{Rotl\_xxx\_yy}(a_2 \oplus \mathrm{AD}_2, 7) = \mathrm{Rotl\_xxx\_yy}(a_2 \oplus (0^{(127)}||1), 7) \ . \tag{52}$$

For the words of b_2 and c_2, the following equalities hold:

$$c_{20} = b_{20} = x_{23} \ , \tag{53}$$
$$c_{21} = b_{21} = x_{20} \ , \tag{54}$$
$$c_{22} = b_{22} = x_{21} \ , \tag{55}$$
$$c_{23} = b_{23} \oplus (0^{(24)}||1||0^{(7)}) = x_{22} \oplus (0^{(24)}||1||0^{(7)}) \ , \tag{56}$$

and so

$$y_2 = (y_{20}, y_{21}, y_{22}, y_{23}) = (x_{20}, x_{21}, x_{22} \oplus (0^{(24)}||1||0^{(7)}), x_{23}) \ . \tag{57}$$

Therefore x_2 and y_2 differ only in the 89-th bit (counting from MSB to LSB).

Proof of Statement 4: Difference Between Words x_3 and y_3. In eq. (38) denote

$$a_3 = (s_3 \lll 32) \oplus ((s_4 \lll 64) \wedge x_0) \ . \tag{58}$$

For x_3 we have

$$\begin{aligned}
x_3 &= \text{Rotl_xxx_yy}(a_3 \oplus (x_1 \ggg 64) \oplus \text{AD}_1, 22) && (59)\\
&= \text{Rotl_xxx_yy}(a_3 \oplus (x_{12}, x_{13}, x_{10}, x_{11}), 22) && (60)\\
&= ((a_{30} \oplus x_{12}) \lll 22, (a_{31} \oplus x_{13}) \lll 22, (a_{32} \oplus x_{10}) \lll 22, (a_{33} \oplus x_{11}) \lll 22) && (61)\\
&= (x_{30}, x_{31}, x_{32}, x_{33}) \ . && (62)
\end{aligned}$$

For y_3 we have

$$\begin{aligned}
y_3 &= \text{Rotl_xxx_yy}(a_3 \oplus (y_1 \ggg 64) \oplus \text{AD}_2, 22) && (63)\\
&= \text{Rotl_xxx_yy}(a_3 \oplus (x_{12}, x_{13}, x_{10}, x_{11} \oplus (1||0^{(31)})) \oplus (0^{(127)}||1), 22) && (64)\\
&= \text{Rotl_xxx_yy}(a_3 \oplus (x_{12}, x_{13}, x_{10}, x_{11} \oplus (1||0^{(31)}) \oplus (0^{(31)}||1)), 22) && (65)\\
&= ((a_{30} \oplus x_{12}) \lll 22, (a_{31} \oplus x_{13}) \lll 22, (a_{32} \oplus x_{10}) \lll 22,\\
&\quad (a_{33} \oplus x_{11} \oplus (1||0^{(31)}) \oplus (0^{(31)}||1)) \lll 22) && (66)\\
&= (x_{30}, x_{31}, x_{32}, x_{33} \oplus (0^{(9)}||11||0^{(21)})) \ . && (67)
\end{aligned}$$

Therefore x_3 and y_3 differ only in the 106-th and 107-th bit (counting from MSB to LSB).

Proof of Statement 5: Difference Between Words x_4 and y_4. In eq. (39) denote

$$a_4 = s_4 \lll w_1 \ . \tag{68}$$

Then x_4 can be expressed as:

$$\begin{aligned}
x_4 &= \text{Rotl_xxx_yy}(a_4 \oplus (x_0 \wedge x_1) \oplus (x_2 \ggg 32) \oplus \text{AD}_1, 13) && (69)\\
&= \text{Rotl_xxx_yy}(a_4 \oplus (x_0 \wedge x_1) \oplus (x_{23}, x_{20}, x_{21}, x_{22}), 13) && (70)\\
&= ((a_{40} \oplus (x_{00} \wedge x_{10}) \oplus x_{23}) \lll 13,\ (a_{41} \oplus (x_{01} \wedge x_{11}) \oplus x_{20}) \lll 13,\\
&\quad (a_{42} \oplus (x_{02} \wedge x_{12}) \oplus x_{21}) \lll 13,\ (a_{43} \oplus (x_{04} \wedge x_{14}) \oplus x_{22}) \lll 13) && (71)\\
&= (x_{40}, x_{41}, x_{42}, x_{43}) \ , && (72)
\end{aligned}$$

and y_4 is expressed as:

$$\begin{aligned}
y_4 &= \text{Rotl_xxx_yy}(a_4 \oplus (y_0 \wedge y_1) \oplus (y_2 \ggg 32) \oplus \text{AD}_2, 13) && (73)\\
&= \text{Rotl_xxx_yy}(a_4 \oplus (x_0 \wedge y_1) \oplus (x_{23}, x_{20}, x_{21}, x_{22} \oplus (0^{(24)}||1||0^{(7)})) \oplus (0^{(127)}||1), 13) && (74)\\
&= ((a_{40} \oplus (x_{00} \wedge x_{10}) \oplus x_{23}) \lll 13,\ (a_{41} \oplus (x_{01} \wedge (x_{11} \oplus (1||0^{(31)}))) \oplus x_{20}) \lll 13,\\
&\quad (a_{42} \oplus (x_{02} \wedge x_{12}) \oplus x_{21}) \lll 13,\\
&\quad (a_{43} \oplus (x_{04} \wedge x_{14}) \oplus x_{22} \oplus (0^{(24)}||1||0^{(6)}||1)) \lll 13) && (75)\\
&= ((a_{40} \oplus (x_{00} \wedge x_{10}) \oplus x_{23}) \lll 13,\ (a_{41} \oplus (x_{01} \wedge (x_{11} \oplus (1||0^{(31)}))) \oplus x_{20}) \lll 13,\\
&\quad (a_{42} \oplus (x_{02} \wedge x_{12}) \oplus x_{21}) \lll 13,\\
&\quad (a_{43} \oplus (x_{04} \wedge x_{14}) \oplus x_{22}) \lll 13 \oplus (0^{(11)}||1||0^{(6)}||1||0^{(13)})) && (76)\\
&= (x_{40}, x'_{41}, x_{42}, x_{43} \oplus (0^{(11)}||1||0^{(6)}||1||0^{(13)})) \ , && (77)
\end{aligned}$$

where x'_{41} differ from x_{41} at most in the first (i.e. most significant) bit with probability 1/2. Therefore x_4 and y_4 differ only in the 108-th and 115-th bit, and the 33-th bit is different with probability 1/2. □

B Derivation of the System of Equations (33) in Sect. 6

Let $X = (x_0, x_1, x_2, x_3, x_4) = \text{StateUpdate}(S^0, AD_1)$, and $Y = (y_0, y_1, y_2, y_3, y_4) = \text{StateUpdate}(S^0, AD_2)$. Let $x_1 \oplus y_1 = x_2 \oplus y_2 = x_3 \oplus y_3 = x_4 \oplus y_4 = \Delta M$. Clearly $x_0 = y_0$.

For $x_1 \oplus y_1$ we derive:

$$\begin{aligned} x_1 \oplus y_1 = {} & \text{Rotl\_xxx\_yy}(s_1 \oplus (s_2 \wedge (s_3 \lll w_0)) \oplus s_4 \oplus \text{AD}_1, b_1) \lll w_3 \oplus \\ & \text{Rotl\_xxx\_yy}(s_1 \oplus (s_2 \wedge (s_3 \lll w_0)) \oplus s_4 \oplus \text{AD}_2, b_1) \lll w_3 \end{aligned} \tag{78}$$

$$\begin{aligned} \text{Rotr\_xxx\_yy}((x_1 \oplus y_1) \ggg w_3, b_1) = {} & s_1 \oplus (s_2 \wedge (s_3 \lll w_0)) \oplus s_4 \oplus \text{AD}_1 \oplus s_1 \oplus \\ & (s_2 \wedge (s_3 \lll w_0)) \oplus s_4 \oplus \text{AD}_2 \end{aligned} \tag{79}$$

$$\text{Rotr\_xxx\_yy}((x_1 \oplus y_1) \ggg w_3, b_1) = \text{AD}_1 \oplus \text{AD}_2 \tag{80}$$

For $x_2 \oplus y_2$ we derive:

$$\begin{aligned} x_2 \oplus y_2 = {} & \text{Rotl\_xxx\_yy}(s_2 \oplus ((s_3 \lll w_0) \wedge (s_4 \lll w_1)) \oplus \\ & (x_0 \ggg w_2) \oplus \text{AD}_1, b_2) \lll w_4 \oplus \\ & \text{Rotl\_xxx\_yy}(s_2 \oplus ((s_3 \lll w_0) \wedge (s_4 \lll w_1)) \oplus \\ & (y_0 \ggg w_2) \oplus \text{AD}_2, b_2) \lll w_4 \end{aligned} \tag{81}$$

$$\begin{aligned} \text{Rotr\_xxx\_yy}((x_2 \oplus y_2) \ggg w_4, b_2) = {} & s_2 \oplus ((s_3 \lll w_0) \wedge (s_4 \lll w_1)) \oplus \\ & (x_0 \ggg w_2) \oplus \text{AD}_1 \oplus s_2 \oplus ((s_3 \lll w_0) \wedge (s_4 \lll w_1)) \oplus (y_0 \ggg w_2) \oplus \text{AD}_2 \end{aligned} \tag{82}$$

$$\text{Rotr\_xxx\_yy}((x_2 \oplus y_2) \ggg w_4, b_2) = \text{AD}_1 \oplus \text{AD}_2 \tag{83}$$

For $x_3 \oplus y_3$ we derive:

$$\begin{aligned} x_3 \oplus y_3 = {} & \text{Rotl\_xxx\_yy}((s_3 \lll w_0) \oplus ((s_4 \lll w_1) \wedge x_0) \oplus (x_1 \ggg w_3) \oplus \\ & \text{AD}_1, b_3) \oplus \text{Rotl\_xxx\_yy}((s_3 \lll w_0) \oplus ((s_4 \lll w_1) \wedge y_0) \oplus \\ & (y_1 \ggg w_3) \oplus \text{AD}_2, b_3) \end{aligned} \tag{84}$$

$$\begin{aligned} \text{Rotr\_xxx\_yy}(x_3 \oplus y_3, b_3) = {} & (s_3 \lll w_0) \oplus ((s_4 \lll w_1) \wedge x_0) \oplus (x_1 \ggg w_3) \oplus \\ & \text{AD}_1 \oplus (s_3 \lll w_0) \oplus ((s_4 \lll w_1) \wedge y_0) \oplus (y_1 \ggg w_3) \oplus \text{AD}_2 \end{aligned} \tag{85}$$

$$\text{Rotr\_xxx\_yy}(x_3 \oplus y_3, b_3) = (x_1 \ggg w_3) \oplus (y_1 \ggg w_3) \oplus \text{AD}_1 \oplus \text{AD}_2 \tag{86}$$

For $x_4 \oplus y_4$ we derive:

$$\begin{aligned} x_4 \oplus y_4 = {} & \text{Rotl\_xxx\_yy}((s_4 \lll w_1) \oplus (x_0 \wedge x_1) \oplus (x_2 \ggg w_4) \oplus \\ & \text{AD}_1, b_4) \oplus \text{Rotl\_xxx\_yy}((s_4 \lll w_1) \oplus (y_0 \wedge y_1) \oplus \\ & (y_2 \ggg w_4) \oplus \text{AD}_2, b_4) \end{aligned} \tag{87}$$

$$\begin{aligned} \text{Rotr\_xxx\_yy}(x_4 \oplus y_4, b_4) = {} & (s_4 \lll w_1) \oplus (x_0 \wedge x_1) \oplus (x_2 \ggg w_4) \oplus \\ & \text{AD}_1 \oplus (s_4 \lll w_1) \oplus (y_0 \wedge y_1) \oplus (y_2 \ggg w_4) \oplus \text{AD}_2 \end{aligned} \tag{88}$$

$$\begin{aligned} \text{Rotr\_xxx\_yy}(x_4 \oplus y_4, b_4) = {} & (x_0 \wedge x_1) \oplus (y_0 \wedge y_1) \oplus (x_2 \ggg w_4) \oplus \\ & (y_2 \ggg w_4) \oplus \text{AD}_1 \oplus \text{AD}_2 \end{aligned} \tag{89}$$

From eq. (80), (83), (86) and (89) we obtain the following system that is equivalent to the system (33) from Sect. 6:

$$\begin{cases} \text{Rotr\_xxx\_yy}((x_1 \oplus y_1) \ggg w_3, b_1) = \text{AD}_1 \oplus \text{AD}_2 \\ \text{Rotr\_xxx\_yy}((x_2 \oplus y_2) \ggg w_4, b_2) = \text{AD}_1 \oplus \text{AD}_2 \\ \text{Rotr\_xxx\_yy}(x_3 \oplus y_3, b_3) = (x_1 \ggg w_3) \oplus (y_1 \ggg w_3) \oplus \text{AD}_1 \oplus \text{AD}_2 \\ \text{Rotr\_xxx\_yy}(x_4 \oplus y_4, b_4) = (x_0 \wedge x_1) \oplus (y_0 \wedge y_1) \oplus (x_2 \ggg w_4) \oplus \\ \qquad\qquad\qquad\qquad (y_2 \ggg w_4) \oplus \text{AD}_1 \oplus \text{AD}_2 \end{cases} \tag{90}$$