Cryptanalysis of JAMBU

Thomas Peyrin[1], Siang Meng Sim[1(✉)], Lei Wang[1], and Guoyan Zhang[1,2,3(✉)]

[1] Nanyang Technological University, Singapore City, Singapore
{thomas.peyrin,wang.lei}@ntu.edu.sg, ssim011@e.ntu.edu.sg
[2] School of Computer Science and Technology, Shandong University, Jinan, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
guoyanzhang@sdu.edu.cn

Abstract. In this article, we analyse the security of the authenticated encryption mode JAMBU, a submission to the CAESAR competition that remains currently unbroken. We show that the security claims of this candidate regarding its nonce-misuse resistance can be broken. More precisely, we explain a technique to guess in advance a ciphertext block corresponding to a plaintext that has never been queried before (nor its prefix), thus breaking the confidentiality of the scheme when the attacker can make encryption queries with the same nonce. Our attack is very practical as it requires only about 2^{32} encryption queries and computations (instead of the 2^{128} claimed by the designers). Our cryptanalysis has been fully implemented in order to verify our findings. Moreover, due to the small tag length of JAMBU, we show how this attack can be extended in the nonce-respecting scenario to break confidentiality in the adaptive chosen-ciphertext model (IND-CCA2) with 2^{96} computations, with message prefixes not previously queried.

Keywords: JAMBU · Authenticated encryption · Cryptanalysis · Confidentiality · Caesar competition

1 Introduction

Authenticated encryption is a very useful cryptographic primitive that provides both privacy and authenticity when sending data. It is a handy component for many security engineers and protocol designers as it avoids for example the classical threat of a misinterpretation of the privacy-only security provided by a simple encryption mode. The encryption algorithm usually takes as input a plaintext P, some public associated data AD, a public nonce value IV, a secret key K, and it outputs a ciphertext C and a tag value T. Conversely, the decryption algorithm usually takes as input a ciphertext C, a tag value T, some public associated data AD, a public nonce value IV, a secret key K, and outputs either the original plaintext P or an error flag if the authentication process is not valid. Using an encryption scheme for the privacy part and a MAC for the authenticity part is a possible way to obtain authenticated encryption, but the goal of the

G. Leander (Ed.): FSE 2015, LNCS 9054, pp. 264–281, 2015.
DOI: 10.1007/978-3-662-48116-5_13

ongoing CAESAR competition [5] is to push to the industry a single primitive providing both properties at the same time, with a single core function, which would potentially permit faster and simpler solutions.

JAMBU is an nonce-based authenticated encryption operating mode proposed by Wu and Huang [14], that can be instantiated with any block cipher. Yet, the submission AES-JAMBU to the CAESAR competition uses AES-128 [6] as internal block cipher. The main advantage of JAMBU mode is its low memory requirement, which places it in the group of lightweight authenticated encryption modes. Indeed, when instantiated with a $2n$-bit block cipher and without counting the memory needed to store the secret key, JAMBU will only require to maintain a $3n$-bit internal state, where classical authenticated encryption modes like OCB [9,11] would require a $6n$-bit internal state or even more. In terms of speed performances, AES-JAMBU is reasonably fast, being about twice slower than AES-CBC [13] (but much slower than OCB since the calls to the internal cipher cannot be parallelized).

The security claims of JAMBU are given in the CAESAR competition submission document [14]. When instantiated with a $2n$-bit block cipher, JAMBU processes plaintext blocks of n bits and eventually outputs an n-bit tag T. When the nonce is not reused, JAMBU is claimed to provide $2n$-bit security for confidentiality and n-bit security for authentication. When the nonce is misused (i.e. several encryptions can be performed with the same nonce), JAMBU is claimed to remain reasonably strong. More precisely, in that scenario, the confidentiality of JAMBU is supposed to be only partially compromised as the authors claim that "*it only leaks the information of the first block or the common prefix of the message*". Regarding authentication in the nonce-misuse scenario, the authors remain vague, only mentioning that "*the integrity of* JAMBU *will be less secure but not completely compromised*".

Our Contribution. In this article, we first describe a very practical attack on JAMBU that breaks its confidentiality claim in the nonce-misuse scenario. More precisely, with only $2^{n/2}$ encryption queries and computing time (which amounts to 2^{32} for AES-JAMBU), we are able to predict the value of a ciphertext block corresponding to a chosen plaintext whose prefix has never been queried to the encryption oracle before, which invalidates the designers' $2n$-bit security claim.

Our attack works by trying to force a zero-difference on the input of one of the internal block cipher calls of JAMBU. Normally, forcing such a collision on a $2n$-bit value should require 2^n computations, but thanks to a divide-and-conquer technique, we are able to divide this event in two subparts, for a total cost of $2^{n/2}$ computations. Having a collision on one of the internal block cipher calls will render this particular JAMBU round totally linear with regards to differences, and will eventually allow us to predict a ciphertext block for the next round.

Then, because of the rather small tag size of JAMBU, we are able to extend our technique to the more interesting case of a nonce-respecting attacker. More precisely, with $2^{3n/2}$ computations (which amounts to 2^{96} for AES-JAMBU), one can break JAMBU's confidentiality in the adaptive chosen-ciphertext model, with message prefixes not previously queried.

We first describe JAMBU authenticated encryption mode in Sect. 2 and then explain our nonce-misuse scenario attack in Sect. 3, while the nonce-respecting attack will be presented in Sect. 4. Finally, in order to confirm our claims, we have implemented the nonce-misuse attack on AES-JAMBU as detailed in Sect. 5. We remark that our techniques will work independently of the cipher instantiating the JAMBU mode, yet in the rest of this article we will focus on AES-JAMBU for ease of description.

2 The JAMBU Authenticated Encryption Scheme

2.1 Description of JAMBU

JAMBU uses a k-bit secret key K and an n-bit public nonce value IV to authenticate a variable length associated data AD and to encrypt and authenticate a variable length plaintext P. It produces a ciphertext C, which has the same bit length with plaintext, and an n-bit tag T.

The encryption process of JAMBU consists of 5 phases as described below: padding, initialization, processing of the associated data, processing of the plaintext, and finalization/tag generation. The computation structure is illustrated in Figs. 1, 2 and 3, where each line represents an n-bit value. We will represent the $3n$-bit internal state of JAMBU by the variables (S_i, R_i) with $S_i = (U_i, V_i)$, where R_i, U_i and V_i are n-bit values. We will denote by E_K the internal cipher using the secret key K.

Padding. First, the associated data AD is padded with 10* padding: a '1' bit is appended to the data, followed by the least number of '0' bits (possibly none) to make the length of the padded associated data become a multiple of n bits. Then, the same padding method is applied to the plaintext.

Initialization. As depicted in Fig. 1, JAMBU uses an n-bit public nonce value IV to initialize the internal state: $S_0 = E_K(0^n \| IV) \oplus (0^{2n-3} \| 101)$, where $\|$ denotes concatenation, and $R_0 = U_0$.

Processing of the Associated Data. The padded associated data is divided into n-bit blocks, and then processed block by block as described in Fig. 1. Note that a single padded block $1||0^{n-1}$ will be processed in the case of an empty associated data string. We omit the details of this phase since it is irrelevant to our attack. Moreover, in the rest of the article we will only use **empty** AD strings, so we get $S_1 = (U_1, V_1) = E_K(S_0) \oplus (1||0^{2n-2}||1)$ and $R_1 = R_0 \oplus U_1$.

Processing of the Plaintext. We denote by p the number of plaintext blocks after padding and $P = (P_1, P_2, \ldots, P_p)$. The plaintext is processed block by block as depicted in Fig. 2. At round i, the internal state is updated with the plaintext block P_i by $S_{i+1} = (U_{i+1}, V_{i+1}) = E_K(S_i) \oplus (P_i||R_i)$ and $R_{i+1} = R_i \oplus U_{i+1}$. The ciphertext block C_i is then computed with $C_i = P_i \oplus V_{i+1}$.

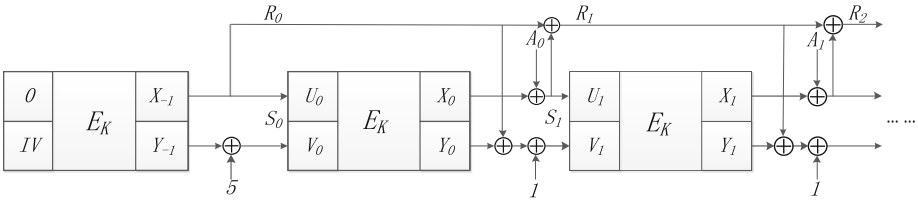

Fig. 1. Initialization and processing of the associated data

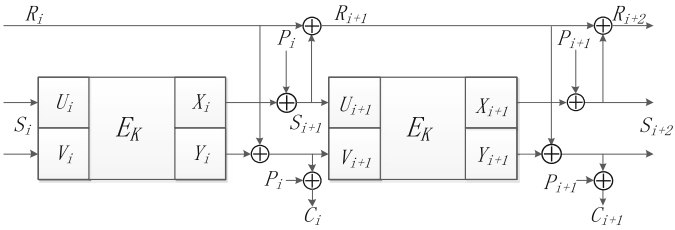

Fig. 2. Processing of the plaintext

Finalization and Tag Generation. When all the plaintext blocks are processed, the final state is (S_{p+1}, R_{p+1}). The authentication tag T is generated with two internal block cipher calls, as depicted in Fig. 3.

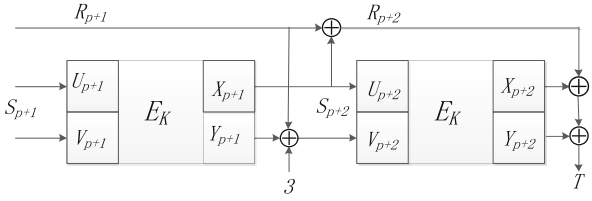

Fig. 3. Finalization and tag generation

2.2 Security Claims

The security claims of `JAMBU` are given in the CAESAR competition submission document [14]. When the nonce is not reused, `JAMBU` is claimed to provide $2n$-bit security for confidentiality and n-bit security for authentication. We note that the type of confidentiality security (i.e. `IND-CPA`, `IND-CCA1` or `IND-CCA2`) is not mentioned by the designers. When the nonce is misused (i.e. several encryptions can be performed with the same nonce), `JAMBU` is claimed to remain reasonably strong. Namely, in that scenario, the confidentiality of `JAMBU` is supposed to be only partially compromised as the authors claim that it only leaks the information of the first block or the common prefix of the message. Regarding authentication in the nonce-misuse scenario, the authors remain vague, only

mentioning that "*the integrity of* JAMBU *will be less secure but not completely compromised*". We summarize in Table 1 the security claims of the CAESAR competition candidate AES-JAMBU where $n = 64$. We remark that as with many authenticated encryption schemes, if verification fails during decryption the new tag and the decrypted plaintext should not be given as output. Moreover it is also important to note that the total amount of message material (plaintext and associated data) that can be protected by a single key is limited to 2^{64} bits for AES-JAMBU.

Table 1. Security claims for AES-JAMBU.

	Confidentiality (bits)	Integrity (bits)
nonce-respecting	128	64
nonce-misuse	128 (except first block or common prefix)	not specified

3 Attack on JAMBU in Nonce-Misuse Scenario

In this section, we analyze JAMBU in the nonce-misuse attack model, where a nonce can be used to encrypt multiple plaintexts. In such a model, JAMBU is an *online* authenticated encryption scheme, namely the i-th ciphertext block is produced before the $i+1$-th plaintext block is read. An inherent property of online authenticated encryption is that common prefix plaintext blocks always produce the same corresponding ciphertext blocks. According to the security claims of JAMBU [14], the only compromised confidentiality security from the nonce-respecting model to the nonce-misuse model is this additional inherent property as becoming an online authenticated encryption in the latter model.

However, we present here a practical attack to distinguish JAMBU from a *random* online authentication encryption, which invalidates the designers' confidentiality security claims of JAMBU in the nonce-misuse model.

3.1 Confidentiality of Online Authenticated Encryption

For an online encryption scheme $(\mathcal{E}_K, \mathcal{D}_K)$ with a key space $\mathcal{K}$, its confidentiality security is usually defined via upper bounding the advantage of all chosen-plaintext distinguishers.[1] We give a brief description as follows, and refer interested readers to [1,7] for the full formal definitions. Let $\mathcal{OAE}$ denote the set of all online authenticated encryption algorithms that have the same block and tag size with $(\mathcal{E}_K, \mathcal{D}_K)$. Let $(\mathtt{OEnc}, \mathtt{ODec}) \xleftarrow{\$} \mathcal{OAE}$ denote an algorithm randomly selected from $\mathcal{OAE}$. Let $\mathbb{D}$ be a distinguisher that interacts with $\mathcal{E}_K$ or Enc, and outputs one bit. Its advantage is defined as:

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{E}}(\mathbb{D}) := \Pr\left[K \xleftarrow{\$} \mathcal{K}, \mathbb{D}^{\mathcal{E}_K} \Rightarrow 1\right] - \Pr\left[(\mathtt{OEnc}, \mathtt{ODec}) \xleftarrow{\$} \mathcal{OAE}, \mathbb{D}^{\mathtt{OEnc}} \Rightarrow 1\right].$$

[1] It has been proven that an authenticated encryption satisfying both IND-CPA and INT-CTXT security notions is also IND-CCA secure [3,4,8].

Then we define $\mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{E}}(t, q, \sigma, \ell) := \max_{\mathbb{D}} \mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{E}}(\mathbb{D})$, where the maximum takes over all distinguishers that run in time t and make q queries, each of length at most ℓ blocks and of total length at most σ blocks.

3.2 Attack Overview

Our attack is based on an observation that we explain below. `JAMBU` maintains a $3n$-bit internal state, but uses only one invocation to a $2n$-bit block cipher E_K to update it per plaintext block. Thus, there are always n state bits per round which are not updated through the strong primitive (i.e. the underlying block cipher). More precisely, every round $S_i = (U_i, V_i)$ is input to the block cipher: $(X_i, Y_i) = E_K(S_i)$. On the other hand, R_i is *linearly* injected into the updated state as $V_{i+1} = Y_i \oplus R_i$. Furthermore, if a pair of plaintexts satisfying $\Delta S_i = 0$ is found, then the state differences in two consecutive rounds are linearly related, i.e., $\Delta V_{i+2} = \Delta R_i$, which will be exploited by our cryptanalysis.

Overall, our attack can be divided into three parts. First the attacker will try to build a special difference structure in the internal state by querying the encryption of a fixed message[2] with several different nonces. Then, using this special differential structure, he will try to recover the values of these internal differences and at the same time force a zero-difference on the input of one of the internal cipher calls. Finally, based on this differential structure that is now fully known and controlled, he will be able to distinguish `JAMBU` from a random online authenticated encryption, and further forge some ciphertext blocks for a message that has never been queried before.

3.3 First Step

For the first step, the attacker picks a random n-bit message block P_1, and asks for the encryption of this message for $2^{n/2}$ distinct nonce values. Since the corresponding ciphertext blocks are also n-bit long, the attacker will have a good chance to observe a collision on the block C_1. We denote IV and IV' the two nonces leading to that collision. One can easily see from Fig. 4 that since no difference is inserted in the block P_1, a collision on C_1 necessarily means that we have a collision on the difference value of the upper branch and lower branch of the internal state (i.e. the difference values in R_1 and in Y_1 are equal). We denote that difference by Δ_R, and we denote the random difference in X_1 by Δ_S. We remark that for this first attack step, we do not need to reuse any nonce value.

3.4 Second Step

In the second step, the goal will be to deduce the value of Δ_S and Δ_R which remain unknown at this moment. In order to achieve this, the attacker will now

[2] We note that it is not necessary that the message is fixed. Just for the simplicity of description, we use a fixed message here and in subsequent sections.

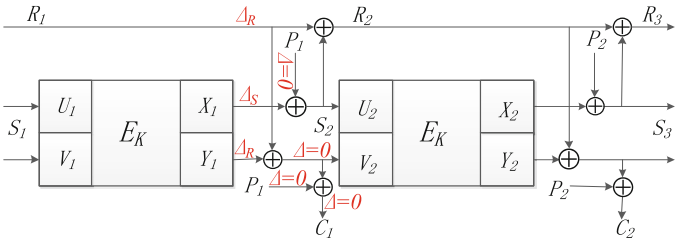

Fig. 4. The first step of the attack.

try to insert a difference in P_1 in a hope that it will be equal to Δ_S. If his choice is right, one can see from Fig. 5 that he will cancel the difference in U_2 and that the difference appearing on the next ciphertext block C_2 (for a plaintext block P_2 without difference) will necessarily be Δ_R. A key observation is that since no difference will be present any more on the input of the incoming block cipher call, the difference on C_2 will remain Δ_R **whatever the choice on the value of** P_1. To summarize, if the attacker adds the difference Δ_S in P_1, then the difference in C_2 will remain the same (i.e. Δ_R) whatever the value of P_1 is. This behavior is what the attacker will use to detect when he makes the right choice for the difference insertion in P_1.

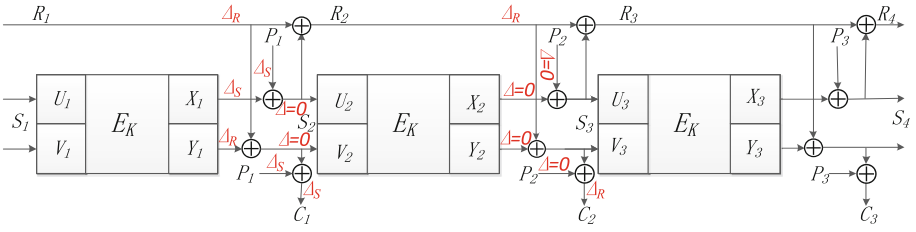

Fig. 5. The second step of the attack.

The detailed procedure to find Δ_S is as follows. Firstly, the attacker constructs two tables as depicted in Fig. 6, each having $2^{n/2}$ three-tuples of one-block plaintexts, such that all pairs created by taking one element from each of these two tables will correspond to all the 2^n possible differences on a n-bit value. More precisely, let $\langle i \rangle$ denote the integer i in a $n/2$-bit binary representation[3]. One table $\mathcal{T}_1$ is $\{ (\langle 0 \rangle \| \langle i \rangle, \langle 1 \rangle \| \langle i \rangle, \langle 2 \rangle \| \langle i \rangle) \}$, where i ranges over all $n/2$-bit values.[4] We denote by $(\langle 0 \rangle \| \langle i \rangle, \langle 1 \rangle \| \langle i \rangle, \langle 2 \rangle \| \langle i \rangle)$ the i-th element of $\mathcal{T}_1$. The other table $\mathcal{T}_2$ is $\{ (\langle j \rangle \| \langle 0 \rangle, \langle 1 \oplus j \rangle \| \langle 0 \rangle, \langle 2 \oplus j \rangle \| \langle 0 \rangle) \}$, where j

[3] Typically n is 64, or 128, i.e. even integers.

[4] Note that the attacker can use any three distinct constants other than $\langle 0 \rangle, \langle 1 \rangle$, and $\langle 2 \rangle$ to construct the tables. Here we use these particular constants just for the simplicity of notations.

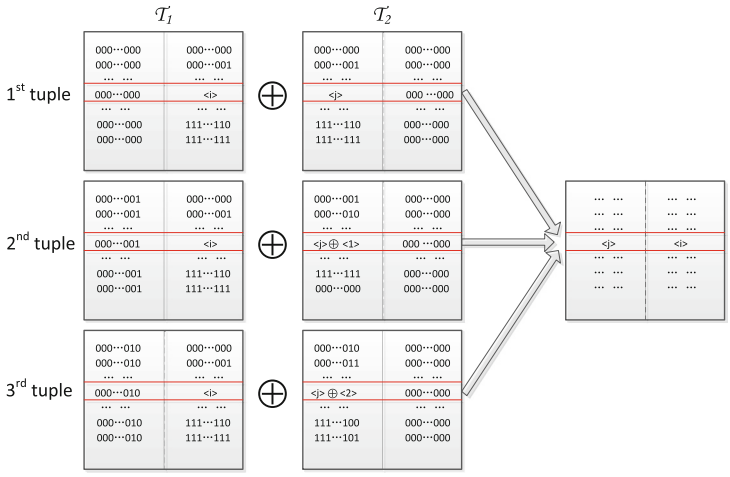

Fig. 6. The tables $\mathcal{T}_1$ and $\mathcal{T}_2$. An example of element pair for difference $\langle j\rangle\|\langle i\rangle$.

ranges over all $n/2$-bit values. Similarly we denote the j-th element of $\mathcal{T}_2$ by $(\langle j\rangle\|\langle 0\rangle,\ \langle 1\oplus j\rangle\|\langle 0\rangle,\ \langle 2\oplus j\rangle\|\langle 0\rangle)$. The pairwise differences between $\mathcal{T}_1$ and $\mathcal{T}_2$ are $\{(\langle 0\rangle\|\langle i\rangle)\oplus(\langle j\rangle\|\langle 0\rangle) = \langle j\rangle\|\langle i\rangle\}$, where i and j independently range over all $n/2$-bit values. Thus, one can see that it covers all the possible differences of one-block n-bit plaintext. In particular, although each element is a 3-tuple of plaintexts and hence each pair consist of three n-bit differences by XORing the corresponding plaintexts, these differences are all equal, i.e., $(\langle 0\rangle\|\langle i\rangle)\oplus(\langle j\rangle\|\langle 0\rangle) = (\langle 1\rangle\|\langle i\rangle)\oplus(\langle 1\oplus j\rangle\|\langle 0\rangle) = (\langle 2\rangle\|\langle i\rangle)\oplus(\langle 2\oplus j\rangle\|\langle 0\rangle)$.

Secondly, the attacker selects a random one-block plaintext P_2. For each element $(\langle 0\rangle\|\langle i\rangle, \langle 1\rangle\|\langle i\rangle, \langle 2\rangle\|\langle i\rangle)$ in table $\mathcal{T}_1$, he uses separately the three plaintext block values as the first block P_1, concatenates them with P_2 as the second block, and makes three encryption queries with the nonce IV to receive the three corresponding ciphertexts. Then, the attacker computes the pairwise differences on the second block of these ciphertexts. In details, let $C[\langle 0\rangle\|\langle i\rangle]_2$, $C[\langle 1\rangle\|\langle i\rangle]_2$ and $C[\langle 2\rangle\|\langle i\rangle]_2$ denote the second ciphertext blocks corresponding to $\langle 0\rangle\|\langle i\rangle\|P_2$, $\langle 1\rangle\|\langle i\rangle\|P_2$ and $\langle 2\rangle\|\langle i\rangle\|P_2$ respectively. The attacker computes the following two n-bit differences and stores them.

$$\Delta C[\langle i\rangle]_1 = C[\langle 1\rangle\|\langle i\rangle]_2 \oplus C[\langle 0\rangle\|\langle i\rangle]_2, \quad \Delta C[\langle i\rangle]_2 = C[\langle 2\rangle\|\langle i\rangle]_2 \oplus C[\langle 0\rangle\|\langle i\rangle]_2.$$

Similarly, for each element of the second table $\mathcal{T}_2$, the attacker makes encryption queries with the above P_2 as the second block and IV' as the nonce to receive the ciphertexts, and then computes the pairwise differences of the second ciphertext blocks, denoted as $(\Delta C'[\langle i\rangle]_1,\ \Delta C'[\langle i\rangle]_2)$. Then he matches the differences to previously stored $\{(\Delta C[\langle i\rangle]_1,\ \Delta C[\langle i\rangle]_2)\}$. Once a matched pair is found, the attacker computes Δ_R and Δ_S from the corresponding plaintexts and ciphertexts as follows

$$\Delta_R = C[\langle 0\rangle \| \langle i\rangle]_2 \oplus C'[\langle j\rangle \| \langle 0\rangle]_2, \quad \Delta_S = (\ \langle 0\rangle \| \langle i\rangle\ ) \oplus (\ \langle j\rangle \| \langle 0\rangle\ ) = \langle j\rangle \| \langle i\rangle.$$

If the attacker does not find a match after running all elements in $\mathcal{T}_2$, he outputs 0.

Now we evaluate the success probability of this step if the attacker interacts with JAMBU. For a pair $(\langle 0\rangle \| \langle i\rangle) \| P_2$ with nonce IV and $(\langle j\rangle \| \langle 0\rangle) \| P_2$ with nonce IV', if $\Delta_S = \langle j\rangle \| \langle i\rangle$, we have that $C[\langle 0\rangle \| \langle i\rangle]_2 \oplus C'[\langle j\rangle \| \langle 0\rangle]_2 = \Delta_R$ as explained in Sect. 3.2. Similarly, we have that $C[\langle 1\rangle \| \langle i\rangle]_2 \oplus C'[\langle 1 \oplus j\rangle \| \langle 0\rangle]_2 = \Delta_R$ and $C[\langle 2\rangle \| \langle i\rangle]_2 \oplus C'[\langle 2 \oplus j\rangle \| \langle 0\rangle]_2 = \Delta_R$. Then, we further deduce that

$$\begin{aligned} & C[\langle 0\rangle \| \langle i\rangle]_2 \oplus C'[\langle j\rangle \| \langle 0\rangle]_2 = C[\langle 1\rangle \| \langle i\rangle]_2 \oplus C'[\langle 1 \oplus j\rangle \| \langle 0\rangle]_2 \\ \Rightarrow \quad & C[\langle 0\rangle \| \langle i\rangle]_2 \oplus C[\langle 1\rangle \| \langle i\rangle]_2 = C'[\langle j\rangle \| \langle 0\rangle]_2 \oplus C'[\langle 1 \oplus j\rangle \| \langle 0\rangle]_2 \\ \Rightarrow \quad & \Delta C[\langle i\rangle]_1 = \Delta C'[\langle j\rangle]_1 \end{aligned}$$

With an identical reasoning, we deduce that $\Delta C[\langle i\rangle]_2 = \Delta C'[\langle j\rangle]_2$. On the other hand, if $\Delta_S \neq \langle j\rangle \| \langle i\rangle$, the pair of the i-th element from $\mathcal{T}_1$ and the j-th element from $\mathcal{T}_2$ will have to satisfy the two n-bit equality conditions randomly, which will happen with probability 2^{-2n}. Since there are in total 2^n such pairs, the probability of faulty positive pairs is negligible. Hence, the attacker gets the correct values of Δ_S and Δ_R with a probability very close to 1.

3.5 Third Step

Finally, in the third and last step, the attacker will choose a random one-block value P_1 such that P_1 and $P_1 \oplus \Delta_S$ have never been queried before as first plaintext block (he can simply keep track of the previously queried P_1 values). Then, he will pick a random value for the second plaintext block P_2 and ask the encryption of the message $(P_1||P_2)$ with the nonce IV. He receives ciphertext blocks C_1 and C_2 from the encryption oracle. Then the attacker asks the encryption of another message $(P_1 \oplus \Delta_S \| P_2)$ with the nonce IV', and receives ciphertext blocks C_1' and C_2'. Then he computes $C_2 \oplus C_2'$, and compares it to Δ_R. If $C_2 \oplus C_2' = \Delta_R$ holds, the attacker outputs 1. Otherwise, the attacker outputs 0.

One can easily evaluate the advantage of the attacker. For JAMBU, he will output 1 with a probability equal to 1. On the other hand, for a random authenticated encryption, he outputs 1 with a probability of 2^{-n}. Therefore, the advantage of the attacker is almost 1.

3.6 Attack and Complexity Summary

To summarize, our attack requires in total about $O(2^{n/2})$ encryption queries and computations, and can be divided into three parts:

- **first step** ($2^{n/2}$ encryption queries and computations): the attacker picks a plaintext block P_1 and queries encryption of this block for $2^{n/2}$ distinct nonces. He keeps the nonce pair (IV, IV') that leads to a collision on the ciphertext block C_1.

- **second step** ($O(2^{n/2})$ encryption queries and computations): the attacker picks a random second plaintext block P_2 and a random $n/2$-bit value I and queries the encryption of the $2^{n/2}$ plaintext blocks $P_1 = (0^{n/2}\|I)$ concatenated with P_2 with nonce IV and the encryption of the $2^{n/2}$ plaintext blocks $P_1 = (I\|0^{n/2})$ concatenated with P_2 with nonce IV'. He repeats the process with a few other constant values instead of $0^{n/2}$ in order to improve the filtering, and he eventually deduces the value of Δ_S by checking which difference applied in P_1 leads to the same difference in C_2 whatever is the choice of I. He directly deduces that this difference on C_2 is actually Δ_R.
- **third step** (2 encryption queries and computations): the attacker picks a random value P_1 such that P_1 and $P_1 \oplus \Delta_S$ have never been queried before, and asks the encryption oracle for the ciphertext corresponding to the message $(P_1||P_2)$ with nonce IV. He receives $(C_1||C_2)$. Then he queries the encryption of $(P_1 \oplus \Delta_S\|P_2)$ with nonce IV' and receive $(C_1'\|C_2')$. Finally he checks if $C_2' \oplus C_2 = \Delta_R$ holds.

We remark that for JAMBU-AES [14], we have $n = 64$ and thus the confidentiality security is only around 32 bits in the nonce-misuse attack model. Thus, our cryptanalysis invalidates the confidentiality claims of the JAMBU designers.

3.7 Extension to a Plaintext-Recovery Attack

Our distinguishing attack can be extended to a more powerful plaintext-recovery attack in a straightforward way. The setting is as follows. Note that our attack is in the chosen-plaintext model, and hence the attacker requires only the encryption algorithm of JAMBU. In other words he is given access to an encryption oracle of JAMBU instantiated with a randomly selected key that is secret to the attacker. He is allowed to query any plaintext of his own choice and gets the corresponding ciphertext. In the end, the attacker is required to choose a new (nonce, ciphertext) pair[5] and to produce a corresponding plaintext for it. If the plaintext is indeed valid and if the prefix to its last block has never been queried before, then the plaintext-recovery attack is said to succeed (the reason of these restrictions is detailed in the discussion on trivial attacks in Sect. 3.8).

The procedure is as follows and it also has three steps. The first two steps are exactly the same as the first two steps of the distinguishing attack detailed in Sects. 3.3 and 3.4, and we adopt the same notations. In the third and last step, the attacker will choose a random value P_1 such that P_1 and $P_1 \oplus \Delta_S$ have never been queried before as first plaintext block under the nonce IV and IV'. Then, he will pick a random value for the second plaintext block P_2 and ask the encryption of the message $(P_1||P_2)$ with the nonce IV. He receives ciphertext blocks C_1 and C_2 from the encryption oracle. Since he knows the value of Δ_R and Δ_S, he will be sure that if he applies the difference Δ_S on P_1 with nonce IV', he

[5] We note that here the attack is only to analyze processing of plaintext/ciphertex of JAMBU, that is confidentiality of encryption, and hence relaxes the setting such that the attacker is not required to provide the tag for his choice of (nonce, ciphertext) pair.

will get difference Δ_S on C_1 and difference Δ_R on C_2. Therefore, he can predict the plaintext $(P_1 \oplus \Delta_S, P_2 \oplus \Delta_R)$ corresponding to ciphertext $(C_1 \oplus \Delta_S || C_2)$ with nonce IV'. Moreover, it is easy to see that $(C_1 \oplus \Delta_S || C_2)$ is not a prefix of any of previous returned ciphertext of the encryption of `JAMBU`, since the first ciphertext block is a permutation of the first plaintext block under the same nonce, and since $P_1 \oplus \Delta_S$ has not been queried before as the first plaintext block under IV' to the encryption oracle. One might argue that P_1 is the first plaintext block and this is included in the security exclusions in the `JAMBU` security claims. However, we have used P_1 for simplicity of description, but the attack remains the same with any amount of random message blocks prepended to P_1.

The complexity of the above plaintext-recovery attacks is also $O(2^{n/2})$ encryption queries and computations. The success probability is almost 1 (we omit the detailed evaluation since it is similar to the distinguishing attack).

3.8 Discussion on Trivial Attacks

Recently, Rogaway claimed a *generic* plaintext-recovery attack on online authenticated encryption in the nonce-misuse setting [10]. His attack adopts divide-and-conquer strategy and recovers the plaintext block by block. In details, the attacker uses the recovered first $i-1$ plaintext blocks as prefix, guesses the i-th plaintext block, and verifies the correctness by sending it to encryption oracle and comparing the received i-th ciphertext block with the i-th target ciphertext block. However, obviously this attack essentially just reveals again the inherent weakness of online authenticated encryption that has been known before and has been also explicitly pointed out by the designer of `JAMBU`: common prefix plaintext blocks produces the same corresponding ciphertext blocks. In particular, the attacker has to query a plaintext to the encryption oracle, then receive a ciphertext that is exactly the same as the target ciphertext, and then output this plaintext as the correct plaintext. As a comparison, in our plaintext recovery setting, we explicitly exclude such rather trivial attacks by restricting that the last block of the target ciphertext (or plaintext) must not share its prefix with any previously returned ciphertext from the encryption oracle.

One can also think of the following trivial distinguishing attack on `JAMBU` and several other CAESAR candidates. For an ideal online authenticated encryption as defined in [1,7], the i-th plaintext block should be input to a random permutation to produce the i-th ciphertext block, where the index of the random permutation is determined by the nonce, the associated data and the first $i-1$ plaintext blocks. On the other hand, for `JAMBU` the i-th plaintext block is simply XORed to an internal state: $C_i = P_i \oplus V_{i+1}$, where the value of V_{i+1} is determined by the nonce, the associated data and the first $i-1$ plaintext blocks. Hence, $\Delta C_i = \Delta P_i$ always holds under the same nonce, the same associated data and the same first $i-1$ plaintext blocks. In details, an attacker queries a nonce IV and a one-block plaintext P_1 to the encryption oracle, and receives a ciphertext C_1. He then queries the same nonce IV and another one-block plaintext P_1' to the encryption, and receives a ciphertext C_1'. If $C_1 \oplus C_1' = P_1 \oplus P_1'$ holds,

the attacker outputs 1. Otherwise, he outputs 0. This distinguishing attack can trivially be extended to a plaintext-recovery attack on single-block ciphertexts.

As a comparison, our attacks reveal a specific weakness of JAMBU: when processing plaintext blocks, it uses only one invocation to a small block cipher ($2n$ bits) to update a larger state ($3n$ bits). Such a design choice obviously favours efficiency, but our attacks imply that there is a greater security compromise to pay than originally expected by the JAMBU designers.

4 Attack on JAMBU in Nonce-Respecting Scenario

In this section, we analyse the confidentiality security of JAMBU in the nonce-respecting scenario. JAMBU claims a $2n$-bit confidentiality security (or 128-bit security for AES-JAMBU) in this setting. However, the claim statement does not contain any specification on the attack model considered (IND-CPA, IND-CCA1 or IND-CCA2). Hence, one may wonder if JAMBU can achieve such a confidentiality security level under all (previously known) attack models[6]. We note that the adaptive chosen-ciphertext security (IND-CCA2) of JAMBU can be trivially broken with 2^n queries by reusing messages with common prefixes (see Sect. 4.4). However, our distinguishing attack works with prefixes not previously queried. Furthermore, our method can be extended to a more powerful plaintext-recovery attack.

4.1 Confidentiality Under an Adaptive Chosen-Ciphertext Attack

For an authenticated encryption scheme $(\mathcal{E}_K, \mathcal{D}_K)$ with a key space $\mathcal{K}$, its confidentiality security under adaptive chosen-ciphertext attacks has been defined in [2], usually referred to as IND-CCA2. Here we provide a brief description, and refer interested readers to [2] for the full formal definition. Let $\mathcal{RO}$ denote a random oracle that has the same output bit length as $\mathcal{E}_K$ on every input plaintext. Let $\mathbb{D}$ be a distinguisher that interacts with $(\mathcal{E}_K, \mathcal{D}_K)$ or $(\mathcal{RO}, \mathcal{D}_K)$, and outputs one bit. Its advantage is defined as:

$$\mathbf{Adv}^{\mathrm{cca2}}_{\mathcal{E}}(\mathbb{D}) := \Pr\left[K \xleftarrow{\$} \mathcal{K}, \mathbb{D}^{\mathcal{E}_K, \mathcal{D}_K} \Rightarrow 1\right] - \Pr\left[K \xleftarrow{\$} \mathcal{K}, \mathbb{D}^{\mathcal{RO}, \mathcal{D}_K} \Rightarrow 1\right].$$

Then we define $\mathbf{Adv}^{\mathrm{cca2}}_{\mathcal{E}}(t, q, \sigma, \ell) := \max_{\mathbb{D}} \mathbf{Adv}^{\mathrm{cca}}_{\mathcal{E}}(\mathbb{D})$, where the maximum is taken over all distinguishers that run in time t and makes q queries, each of length at most ℓ blocks and of total length at most σ blocks. The distinguisher must not make two queries with the same nonce to the encryption oracle that is $\mathcal{E}_K$ or $\mathcal{RO}$. Moreover, we assume the distinguisher does not query the outputs from one oracle to the other oracle. Namely, he does not query the received ciphertext from $\mathcal{E}_K$ or $\mathcal{RO}$ to $\mathcal{D}_K$, and does not query the received plaintext from $\mathcal{D}_K$ to $\mathcal{E}_K$ or $\mathcal{RO}$. These assumptions aim at preventing trivial distinguishing attacks.

[6] Yet we trivially observe that JAMBU can only achieve 2^n confidentiality security in the IND-CCA3 model [12] (and not the expected 2^{2n}), due to its n-bit tag size.

4.2 Distinguishing Attack

We notice that JAMBU uses an n-bit tag. Therefore, one can always obtain the corresponding plaintext for a ciphertext of his own choice from the decryption oracle by making at most 2^n queries, i.e., by exhaustively guessing the tag value. Based on this observation, we can transform the distinguishing attack in the nonce-misuse setting detailed in Sect. 3 to a distinguishing attack in nonce-respecting setting, with a complexity increase by a factor 2^n and hence with a total complexity of $2^{3n/2}$, which is lower than the $2n$-bit security one might expect.

In details, the attack in the nonce-misuse setting consists of three steps, and the repeating nonces requirement happens in steps 2 and 3. Thus, we will mainly modify these two steps. We adopt the same notation as Sect. 3.

First Step. The procedure is exactly the same as before. For the plaintext P_1, its ciphertext is denoted as C_1 under the nonce IV and as C_1' under the nonce IV'. Then we denote $V[IV]_2 = P_1 \oplus C_1$ and $V[IV']_2 = P_1 \oplus C_1'$.

Second Step. Firstly, the attacker constructs tables $\mathcal{T}_1$ and $\mathcal{T}_2$ as before. Secondly, he selects a random one-block ciphertext block C_2. For each element $(\langle 0\rangle\|\langle i\rangle, \langle 1\rangle\|\langle i\rangle, \langle 2\rangle\|\langle i\rangle)$ in table $\mathcal{T}_1$, he executes a similar procedure to interact with the decryption oracle for each of $\langle 0\rangle\|\langle i\rangle$, $\langle 1\rangle\|\langle i\rangle$ and $\langle 2\rangle\|\langle i\rangle$. Here we use $\langle 0\rangle\|\langle i\rangle$ as an example to describe this procedure. The attacker computes $V[IV]_2 \oplus (\langle 0\rangle\|\langle i\rangle)$ as the first ciphertext block, concatenates it with C_2 as the second block, and queries the constructed two-block ciphertext to the decryption oracle with the nonce IV and with a random selected tag value. If the decryption oracle returns a failure symbol $\perp$, the attacker changes the tag to a new value, and makes a decryption query with the same nonce and the same ciphertext. He will repeat such decryption queries by exhaustively trying new tag values until the decryption oracle returns a plaintext instead of $\perp$. In the returned plaintext, it is easy to get that the first block is $\langle 0\rangle\|\langle i\rangle$, and we denote its second block as $P[\langle 0\rangle\|\langle i\rangle]_2$. Similarly, we define notations $P[\langle 1\rangle\|\langle i\rangle]_2$ and $P[\langle 2\rangle\|\langle i\rangle]_2$ for the second plaintext block corresponding to $\langle 1\rangle\|\langle i\rangle$ and $\langle 2\rangle\|\langle i\rangle$ respectively. Once a plaintext obtained, the attacker computes the pairwise differences of the second plaintext blocks as follows:

$$\Delta P[\langle i\rangle]_1 = P[\langle 1\rangle\|\langle i\rangle]_2 \oplus P[\langle 0\rangle\|\langle i\rangle]_2, \quad \Delta P[\langle i\rangle]_2 = P[\langle 2\rangle\|\langle i\rangle]_2 \oplus P[\langle 0\rangle\|\langle i\rangle]_2.$$

For each element of the other table $\mathcal{T}_2$, the attacker makes similar decryption queries, but using IV' as nonce and $V[IV']_2$ to compute the first ciphertext blocks. We denote the computed pairwise differences of the second plaintext blocks as $(\Delta P'[\langle j\rangle]_1, \Delta P'[\langle j\rangle]_2)$. The attacker matches the differences to previously stored $\{(\Delta P[\langle i\rangle]_1, \Delta P[\langle i\rangle]_2)\}$. Once a matched pair is found, the attacker computes Δ_R and Δ_S from the corresponding plaintexts and ciphertexts as follows:

$$\Delta_R = P[\langle 0\rangle\|\langle i\rangle]_2 \oplus P'[\langle j\rangle\|\langle 0\rangle]_2, \quad \Delta_S = (\langle 0\rangle\|\langle i\rangle) \oplus (\langle j\rangle\|\langle 0\rangle) = \langle j\rangle\|\langle i\rangle.$$

If no match is found after trying all elements in $\mathcal{T}_2$, the attacker outputs 0.

Third Step. The attacker selects a random one block C_1 such that C_1 and $C_1 \oplus \Delta_S$ have not been queried before as a first block of ciphertext under the nonces IV and IV'. Then, he selects another random block C_2. Firstly, the attacker makes queries $C_1 \| C_2$ to the decryption oracle with the nonce IV by exhaustively guessing the tag until he receives the plaintext, where the second plaintext block is denoted as P_2. Secondly, the attacker makes queries $C_1 \oplus \Delta_S \| C_2$ to the decryption oracle with the nonce IV' by exhaustively guessing the tag until he receives the plaintext, where the second plaintext block is denoted as P_2'. Finally, he computes $\Delta P_2 = P_2 \oplus P_2'$, and compares it to Δ_R. If $\Delta P_2 = \Delta_R$, the attacker outputs 1. Otherwise, he outputs 0.

The overall complexity is dominated by step 2, which is upper bounded by $O(2^{3n/2})$ (or 2^{96} for `AES-JAMBU`). The advantage of the distinguisher is almost 1 (we omit the detailed evaluation since it is similar with that of the attacks in previous sections).

4.3 Extension to a Plaintext-Recovery Attack

The plaintext-recovery attack setting is as follows. The attacker is given access to both encryption and decryption oracles of `JAMBU` instantiated with a randomly selected key that is secret to the attacker. He is allowed to make encryption and decryption queries of his own choice. Note that he must not make two encryption queries with the same nonce. In the end, the attacker is required to choose a nonce and a ciphertext (where the last block of the ciphertext must not have the same prefix than the last blocks of any previously outputted or queried ciphertext under the same nonce) and to produce a corresponding plaintext for it. If the plaintext is indeed valid, the plaintext-recovery attack is said to succeed.

The attack procedure is similar with that of distinguishing attacks from Sect. 4.2. The first two steps are exactly the same, and we adopt the same notations. In the third and last step, the attacker will choose a random one-block value C_1 such that C_1 and $C_1 \oplus \Delta_S$ have never been outputted as the first ciphertext block from the encryption oracle and have never been queried to the decryption oracle as first ciphertext block under the nonce IV and IV'. Then, he will pick a random value for the second plaintext block C_2 and interact with the decryption oracle to receive the plaintext $P_1 \| P_2$ of the ciphertext $(C_1 || C_2)$ under the nonce IV. Since he knows the value of Δ_R and Δ_S, he will be sure that if he applies the difference Δ_S on P_1 with nonce IV', he will get difference Δ_S on C_1 and difference Δ_R on C_2. Therefore, he can predict the plaintext $(P_1 \oplus \Delta_S, P_2 \oplus \Delta_R)$ corresponding to ciphertext $(C_1 \oplus \Delta_S || C_2)$ with nonce IV'.

The complexity of the above plaintext-recovery attacks is $O(2^{3n/2})$ encryption queries and computations (or 2^{96} for `AES-JAMBU`), and its success probability is almost 1 (we omit the detailed evaluation, since it is similar to the distinguishing attack).

4.4 Discussion on Trivial Attacks

In the nonce-respecting scenario, although the attacker cannot make two encryption queries with the same nonce, he is allowed to repeat nonces during the interaction with the decryption oracle. Hence, if he makes more than 2^n decryption queries, he will obtain more than one pair of plaintext and ciphertext under the same nonce. As a result, this leads to several trivial attacks (similar to the trivial attacks on `JAMBU` in the adaptive chosen-ciphertext attack model described in Sect. 3.8). For example, the attacker can interact with the decryption oracle to receive a plaintext P_1 for nonce IV and a one-block ciphertext C_1, and then interact with the encryption oracle to receive a ciphertext C_1' for a random one-block plaintext P_1' with the same nonce IV. Finally he checks if $P_1 \oplus C_1 = P_1' \oplus C_1'$ holds. We refer to Sect. 3.8 for more discussions on trivial attacks on `JAMBU`.

5 Implementation of the Attack

We have implemented the attack on `AES-JAMBU` for the nonce-misuse scenario as described in Sect. 3 and we have verified the special differential structure from Fig. 5. For simplicity, the associated data was set to be empty, and the 128-bit key was set to `0x100f0e0d0c0b0a090807060504030201`.

5.1 Results of the Attack

In the first step of the attack, we chose a random 64-bit plaintext P_1 and asked for encryption under different nonce values. With 2^{32} encryption queries, we found a collision on a pair of ciphertexts C_1, C_1' with a pair of nonce values IV, IV' (see Table 2).

Table 2. First step of the attack

K :	01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10
IV :	b1 ef 89 a0 4e 21 30 bd
IV' :	10 5a 1f 5b 34 49 1e 5c
P_1 :	7f 95 77 ca 09 77 a8 a5
C_1 :	2d 2b 58 18 fa f5 af f1
C_1' :	2d 2b 58 18 fa f5 af f1

With this pair of nonce values, we proceeded to the second step of the attack, P_2 being set to zero for simplicity. We constructed the tables $\mathcal{T}_1$ and $\mathcal{T}_2$ and by matching the differences in the second block of ciphertexts, we obtained the values of Δ_S and Δ_R. Table 3 shows the first tuple of the pair of plaintexts and ciphertexts tables with the matching difference.

Table 3. Second step of the attack

$\langle j\rangle \| \langle 0\rangle$ ||	P_2	: 60 28 6d 74 00 00 00 00 00 00 00 00 00 00 00 00
	$C'[\langle j\rangle \| \langle 0\rangle]_2$	: af 45 56 9e 26 c6 7e d0
$\langle 0\rangle \| \langle i\rangle$ ||	P_2	: 00 00 00 00 93 47 1e 92 00 00 00 00 00 00 00 00
	$C[\langle 0\rangle \| \langle i\rangle]_2$	: 73 79 44 54 a7 b4 5b 4c
	Δ_S	: 60 28 6d 74 93 47 1e 92
	Δ_R	: dc 3c 12 ca 81 72 25 9c

In the third step, we chose a random 128-bit plaintext $(P_1\|P_2)$ and asked for its encryption with nonce IV. Upon receiving the ciphertext $(C_1\|C_2)$, we deduced the ciphertext $(C_1^D\|C_2^D) = (C_1 \oplus \Delta_S\|C_2 \oplus \Delta_R)$ for the plaintext $(P_1 \oplus \Delta_S\|P_2)$ with nonce IV' without querying it to the encryption oracle. Finally, we checked that by asking for the encryption of the plaintext $(P_1 \oplus \Delta_S\|P_2)$ with nonce IV', the ciphertext $(C_1'\|C_2')$ obtained is indeed what we had deduced (as can be seen from Table 4).

Table 4. Third step of the attack

IV	: b1 ef 89 a0 4e 21 30 bd
$P_1\|P_2$	: 95 d9 43 9e 0b 4d 6d 27 6a ba db 0a 12 f8 13 45
$C_1\|C_2$	: c7 67 6c 4c f8 cf 6a 73 6b 05 9b c6 fc e6 7a ee
Δ_S	: 60 28 6d 74 93 47 1e 92
Δ_R	: dc 3c 12 ca 81 72 25 9c
$C_1^D\|C_2^D$	: a7 4f 01 38 6b 88 74 e1 b7 39 89 0c 7d 94 5f 72
IV'	: 10 5a 1f 5b 34 49 1e 5c
$P_1 \oplus \Delta_S\|P_2$	: f5 f1 2e ea 98 0a 73 b5 6a ba db 0a 12 f8 13 45
$C_1'\|C_2'$	: a7 4f 01 38 6b 88 74 e1 b7 39 89 0c 7d 94 5f 72

The codes for the attack on `AES-JAMBU` are included in the supporting document, they are separated in two main codes - Step 1 and Step 2 of the attack, AES-NI is used for running `AES-JAMBU`.

5.2 Running Time of the Attack

For the first step of the attack, it took about 3.7 h and 36 GB of memory to find a collision. While for the second step of the attack, it took about 8.8 h and 320 GB to find Δ_S and Δ_R.

For the second step of the attack, one can do a trade-off between the computation time and memory requirement. For instance, instead of constructing

tables of 2^{32} elements, one can construct tables of 2^{30} (or 2^{28} respectively) elements and the computation time takes about 2.2 h (or 0.5 h respectively) and 80 GB (or 20 GB respectively) of memory. However, in this case, one would have to guess the 2 (or 4 respectively) most significant bits of the difference values i and j. Hence, by repeating the attack procedure 16 times (or 256 times respectively), the value of Δ_S and Δ_R can be recovered by enumerating all the possible most significant bits values.

Conclusion

In this article, we have proposed a cryptanalysis of the confidentiality of JAMBU in both the nonce-misuse and nonce-respecting models. Namely, we have shown that one can break confidentiality in the nonce-misuse scenario with 2^{32} computations and queries, while having access to only the encryption oracle. For the nonce-respecting, we show that our attack can be extended to break confidentiality security of JAMBU with 2^{96} computations and queries in the adaptive chosen-ciphertext attack model, with message prefixes not previously queried.

It would be an interesting future work to study how JAMBU could be patched to resist these attacks. We believe that one simple possibility would be to output P_{i-1} instead of P_i during round i (while keeping the insertion of P_i in the internal state). This would probably prevent our attack since the last block of the distinguishing plaintext/ciphertext pair would have the exact same prefix than the last block of previously queried pairs.

Acknowledgements. The authors would like to thank the JAMBU designers (Hongjun Wu and Tao Huang), Tetsu Iwata and the anonymous referees for their helpful comments. The authors are supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

References

1. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 424–443. Springer, Heidelberg (2013)
2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th Annual Symposium on Foundations of Computer Science, FOCS 1997, Miami Beach, Florida, USA, October 19–22, 1997, pp. 394–403. IEEE Computer Society (1997)
3. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
4. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. J. Cryptol. **21**(4), 469–491 (2008)
5. Bernstein, D.: CAESAR Competition. http://competitions.cr.yp.to/caesar.html

6. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography. Springer, Heidelberg (2002)
7. Fleischmann, E., Forler, C., Lucks, S.: McOE: a family of almost foolproof on-line authenticated encryption schemes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 196–215. Springer, Heidelberg (2012)
8. Katz, J., Yung, M.: Complete characterization of security notions for probabilistic private-key encryption. In: Yao, F.F., Luks, E.M. (eds.) Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21–23, 2000, pp. 245–254. ACM, Portland (2000)
9. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
10. Rogaway, P.: Let's not Call It MR (2014). http://web.cs.ucdavis.edu/rogaway/beer.pdf
11. Rogaway, P., Bellare, M., Black, J.: OCB: a block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. **6**(3), 365–403 (2003)
12. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)
13. Frankel, S., Glenn, R., Kelly, S.: The AES-CBC Cipher Algorithm and Its Use with IPsec. Network Working Group, RFC 3602, September 2003
14. Wu, H., Huang, T.: JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU (v1). Submitted to the CAESAR competition, March 2014