Security Proof of JAMBU under Nonce Respecting and Nonce Misuse Cases

Geng Wang*, Haiyang Zhang and Fengmei Liu

Science and Technology on Information Assurance Laboratory, Beijing, 100072, P.R. China

Abstract. JAMBU is an AEAD mode of operation which entered the third round of CAESAR competition. However, it does not have a security proof like other modes of operation do, and there was a cryptanalysis result that has overthrown the security claim under nonce misuse case by the designers. In this paper, we complement the shortage of the scheme by giving security proofs of JAMBU both under nonce respecting case and nonce misuse case. We prove that JAMBU under nonce respecting case has a slightly lower security than the birthday bound of n bits, and JAMBU under nonce misuse case has a tight security bound of $n/2$ bits.

Keywords: JAMBU, CAESAR Competition, Provable Security, Nonce-Misuse Resistance

1 Introduction

Authenticated encryption, or usually known as authenticated encryption with associated data (AE or AEAD for short), which was formalized in [3, 17], is a cryptographic primitive that can protect confidentiality and integrity at the same time. AEAD takes as input a public nonce IV, public associated data AD, plaintext P, and a key K, outputs the ciphertext C and a tag T while encryption, and while decryption, K, IV, AD, C and T are inputs, if the tag T is valid, returns P, otherwise an error symbol $\perp$.

In 2013, the international cryptologic research community announced a new competition for authenticated encryption called CAESAR, and in August 2016, 15 candidates were elected into the third round, including JAMBU by Wu and Huang [25]. JAMBU (originally AES-JAMBU) is a block-cipher mode of operation, which is a primary method for implementing authenticated encryption. Among other AEAD schemes using modes of operation in CAESAR, JAMBU is designed for lightweight applications. It is not as fast as the parallelizable schemes such as OCB [20] and OTR [12], but it is inverse-free, using only X-OR operations, and has a lower state size in the cost of a shorter nonce and tag length [25]. JAMBU adopts an underlying block-cipher with $2n$-bit block length and k-bit key length, along with n-bit nonce, and outputs n-bit tag. It

* E-mail: cnpkw@126.com

has only $3n$-bit state, which memory requirement is among the least of CAESAR candidates.

Initial vector (IV), or usually called nonce to show its non-repeatedness, has been important in symmetric key cryptography since the invention of CBC mode. The importance of nonce has been discussed by earlier researchers, especially in the terms of AEAD [4, 18]. Each nonce was supposed to be used only once, but due to various reasons including incorrect implementation, resource limitation, loss of stored nonce data, etc, it is possible that an encryption algorithm returns two ciphertexts with a same nonce, which is often called *nonce misuse*. Most earlier AEAD modes of operation were not designed to support nonce misuse. For example, in GCM mode of operation which is widely adopted as a standard [14], the security would be completely broken if nonce could be reused. But later, especially as the CAESAR competition went on, the community and AEAD designers were divided into two groups: some of them believe nonce should never be reused, and others believe that nonce misuse is inevitable, so an AEAD scheme should at least provide some security when the user repeats a nonce.

The idea of AE with nonce misuse security, has first been introduced with the term deterministic AE [22] or misuse-resistance AE [21]. Researchers also provided some modes of operation which support nonce misuse resistance, such as [10, 23, 7]. However, such security notion requires that there is no information leakage even when nonce is reused, which is sometimes too strong. So a weaker notion of online AE has been studied, often called online nonce misuse resistance AE [6, 1, 8]. An AE scheme is called online, if each output block is only related to its previous input blocks. A perfectly secure online scheme should only leak the common prefix of the message. In the CAESAR competition, there are also some online nonce misuse resistance schemes, for example COLM [2] that has entered the third round.

Although the necessity of nonce misuse security is still under controversy, it is indeed useful for lightweight applications. For protecting confidentiality and integrity in a resource restrained device such as IoT, RFID card, etc, it is not always possible for storing and managing fresh nonce, and sometimes it requires additional synchronous protocol which might be costly. But if the scheme is nonce-misuse resistance, even only to a small degree, then any random number could be used as a nonce, which will simplify the implementation a lot.

In the first version of JAMBU proposal [24], the designers claimed that JAMBU leaks only the common prefix of the message when nonce is reused. However, JAMBU with nonce misuse had later been analyzed by Peyrin et al [16], and they showed that there is an attack with $O(2^{n/2})$ queries on JAMBU with nonce misuse. The designers acknowledged their work. In their latest document [25], the designers gave a proof on the authenticity of JAMBU, but there are still no results on privacy. However, they believe that JAMBU can achieve some security under nonce misuse, although not as they originally claimed. If JAMBU could be proved to have an $n/2$-bit security under nonce misuse case, although not full security, it could still bring great advantage since JAMBU is designed for lightweight usage.

Provable security is an important method in the research of both public key and symmetric key cryptography. In symmetric key cryptography, provable security is usually applied to modes of operation, which security is reduced to the security of the underlying block cipher. The examples are security proofs to OCB [19] and GCM [15]. Although not necessary, a security proof is often considered a great advantage when evaluating a mode of operation. Most of the CAESAR submissions which are modes of operation had given their security proofs. However, the designers did not give their security proof on JAMBU, and this devaluate their security claims compared to other schemes such as CLOC/SILC [9], which shares the same lightweight feature with JAMBU. In this paper, we shall give security proofs for JAMBU under both nonce respecting and nonce misuse cases, so that the security of JAMBU could be further ensured.

The mainly method used for proving security for modes of operation, which we shall also use in this paper, is called game-playing proof [5]. A game-playing proof is used to estimate the maximal probability for the adversary to distinguish a cryptographical algorithm and an ideal model (such as the random oracle model). It is done by a construction of several interactive games GAME1 to GAMEn, where GAMEn is the original algorithm and GAME1 is the ideal model. Each two if them, say GAMEi and GAMEi+1 have only small differences, and the maximal probability for the adversary to distinguish between GAMEi and GAME$i + 1$ can be calculated. Thus, the upper bound on distinguishing probability is the sum of all these probabilities. By defining an event (usually called a *bad* event), where the two games are indistinguishable unless the event occurs, the distinguishing probability turns into calculating the probability of the event.

Unlike nonce respecting security, there is no common way to define nonce misuse security. In [8], the authors gave out various security notions, one is called OAE1d, which a repeated nonce leaks not only the common prefix, but the XOR between plaintext and ciphertext of the next block, which captured the security notion of JAMBU to some degree. However, there was no formalization and further analysis in their paper. So rather than adopting their notions, we shall redefine the nonce misuse security for JAMBU ourselves.

In this paper, we prove that JAMBU indeed can be proven to have a nearly n-bit security under nonce respecting case, which is close to its birthday bound, and for nonce-misuse case, we define a JAMBU-like online oracle as its ideal security model, and prove that there is a $n/2$-bit security under nonce misusing case using this model.

The paper organized as follows. In section 2, we simply introduce the JAMBU scheme. In section 3, we make preparations for the proof by introducing notations and security models. In section 4, we prove the security of JAMBU under nonce respecting case. In section 5, we prove the security of JAMBU under nonce misuse case. And finally in section 6 we draw the conclusion.

2 The JAMBU AE Scheme

Before we go on to the security proof, we first introduce the structure of JAMBU. As a block cipher mode of operation, JAMBU uses an underlying block-cipher with k-bit key and $2n$-bit block length, takes n-bit nonce IV, arbitrary length of associated data AD and plaintext P as input, generates ciphertext C of the same length as P, and n-bit authentication tag T. JAMBU has $3n$-bit internal states U, V and R, each of n-bit. In their submission, the designers denote the states before a block-cipher call by U, V, R and after a block-cipher call by X, Y, R, we adopt this notation in our discussion. We write E_K as the underlying block cipher.

A JAMBU encryption consists of five steps:

(1) Padding. AD and P are done with a 10^* padding. For associated data, a '1' bit is padded followed by the least number of '0' bits to make the length of padded associated data a multiple of n-bit. Then the same padding method is applied to the plaintext.

(2) Initialization. For the n-bit nonce IV, the state $U_{-1}\|V_{-1}$ is set to $0^n\|IV$, and R_{-1} is set to 0^n. Then, set $X_{-1}\|Y_{-1} \leftarrow E_K(U_{-1}\|V_{-1})$, $U_0\|V_0 \leftarrow X_{-1}\|Y_{-1} \oplus 5$, $R_0 \leftarrow U_0$, $\|$ denotes concatenation. Note that 5 is written as a binary string $0^{2n-3}101$, so are other numbers below.

(3) Processing of the associated data. For the padded associated data (note that if there is no associated data, padding made it into at least 1 block), it is divided into h blocks $AD[0], ..., AD[h-1]$, and processed as follows:

For $i = 0$ to $h-1$, update the states:

$X_i\|Y_i \leftarrow E_K(U_i\|V_i)$, $U_{i+1} \leftarrow X_i \oplus AD[i]$, $V_{i+1} \leftarrow Y_i \oplus R_i \oplus 1$, $R_{i+1} \leftarrow R_i \oplus U_{i+1}$, $AD[i]$ is the i-th AD block.

The initialization and processing of the associated data step is shown in figure 1.

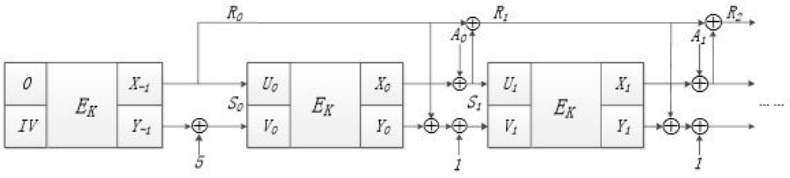

Fig. 1: Initialization and AD processing

(4) Processing of the plaintext. For the padded plaintext, it is divided into p blocks $P[0], ..., P[p-1]$, and processed as follows:

For $i = 0$ to $p-1$ (note that we reset i to 0, and $U_h\|V_h$ in (3) becomes $U_0\|V_0$), update the states:

$X_i\|Y_i \leftarrow E_K(U_i\|V_i)$, $U_{i+1} \leftarrow X_i \oplus P[i]$, $V_{i+1} \leftarrow Y_i \oplus R_i$, $R_{i+1} \leftarrow R_i \oplus U_{i+1}$, output $C[i] \leftarrow P[i] \oplus V_{i+1}$, $P[i]$ is the i-th plaintext block.

For the last ciphertext block $C[p-1]$, truncate it into the same length as the last plaintext block before padding. For example, if plaintext length is a multiple of n (in this case, a full padding block is added), then the last ciphertext block is simply ignored.

The processing of the plaintext step is shown in figure 2.

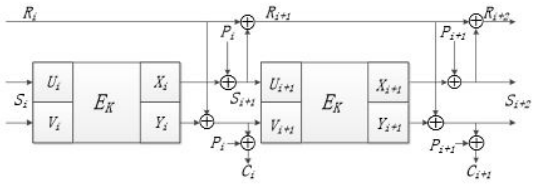

Fig. 2: Plaintext processing

(5) Finalization and tag generation. It process as follows:

$X_p \| Y_p \leftarrow E_K(U_p \| V_p)$, $U_{p+1} \leftarrow X_p$, $V_{p+1} \leftarrow Y_p \oplus R_p \oplus 3$, $R_{p+1} \leftarrow R_p \oplus U_{p+1}$, $X_{p+1} \| Y_{p+1} \leftarrow E_K(U_{p+1} \| V_{p+1})$, output $T \leftarrow X_{p+1} \oplus Y_{p+1} \oplus R_{p+1}$.

The finalization and tag generation step is shown in figure 3.

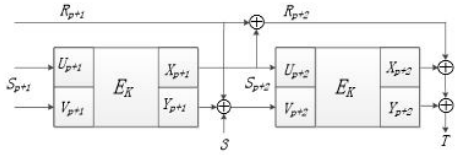

Fig. 3: Finalization and tag generation

In a JAMBU decryption, first do the padding, initialization and AD processing step. Then, generate the state V_1, use it to recover $P[0]$, and use $P[0]$ the same way as the plaintext processing step to generate V_2, then recover $P[1]$, etc. After generate a tag T', compare T' with T, if $T' = T$, output the plaintext, otherwise output $\perp$.

3 Preliminaries

3.1 Notations

We shall use $\|$ as the operator for concatenation of two strings, for example, $0\|1$ is the string 01. ε denotes an empty string. For a string s, we write $s|_{i,j}$ to be the substring of s from the i-th bit to the $j-1$-th bit, for example,

$01100|_{2,4} = 10$. If $i = 0$, we also write it as $s|_j$, which is the first j bits of s. We also let $s[i] = s|_{ni,n(i+1)}$ be the i-th block of s, n is the block size of JAMBU. $x \xleftarrow{\$} X$ means that x is randomly chosen from a set X.

In the discussion below, we suppose that the padding procedure has been done, and ignore the truncation on ciphertext. That means each input/output can be divided into n-bit blocks, and we write H for $IV\|AD[0]\|..\|AD[h-1]$, where h is the number of AD blocks, and P for $P[0]\|...\|P[p-1]$, p is the number of plaintext blocks, CT for $C[0]\|...\|C[p-1]\|T$.

3.2 Security Model

We first define a JAMBU-like online oracle for the ideal security, which will be used later in the proof.

Definition 3.1. *A JAMBU-like online oracle* $\mathcal{O} : (\{0,1\}^n)^+ \times (\{0,1\}^n)^+ \to (\{0,1\}^n)^+$ *is defined as: for the* i*-th query* (H_i, P_i)*, which* H_i *is a* $n(h_i+1)$ *bits binary string (means header), and* P_i *a* np_i *bits binary string (means plaintext),* $\mathcal{O}(H_i, P_i)$ *returns:*

(1) If there exists $j < i$ *such that* $H_j = H_i$, $P_j = P_i$, *returns* $O(H_j, P_j)$*;*

(2) If (1) is not satisfied, and there exists $j < i$ *and* $r \geq 0$ *such that:*

(a) $H_j = H_i$, $P_j|_{nr} = P_i|_{nr}$, $t < p_i$ *and* $t < p_j$*;*

(b) There is no $k < i$ *(including* $k = j$*) such that* $H_k = H_i$, $P_k|_{n(r+1)} = P_i|_{n(r+1)}$, $r+1 < p_i$ *and* $r+1 < p_k$*;*

(c) There is no $k < j$ *such that* $H_k = H_i$, $P_k|_{nr} = P_i|_{nr}$, $r < p_i$ *and* $r < p_k$*;*

Then returns $O(H_j, P_j)|_{n(r+1)}\|x$, *which* $x \xleftarrow{\$} \{0,1\}^{n(p_i - r)}$*;*

(3) Otherwise, returns a random string of $n(p_i+1)$ *bits.*

Condition (2) means that for (H_i, P_i), if among all previous queries with the same header, (H_j, P_j) is the first one that P_j has the most common proper prefix blocks with P_i of r blocks (each of n bits), then return the first $n(r+1)$ bits of $\mathcal{O}(H_j, P_j)$ concatenates with a random string of $n(p_i - r)$ bit.

It is easy to see that, if a JAMBU-like online oracle has never been queried twice with a same header, the condition (2) will never be triggered, and it performs like a random oracle (despite the restriction on input and output length). But when nonce could be reused, such oracle has the "onlineness" property, that the first $n(r+1)$ bits output of $\mathcal{O}(H, P)$ is determined only by H and the first nr bits of P. We prove this property below.

Theorem 3.1. *Let* (H_a, P_a) *and* (H_b, P_b) *be two queries of a JAMBU-like online oracle* $\mathcal{O}$*. If* $H_a = H_b$ *and there is some* $r \geq 0$, $r < p_a$ *and* $r < p_b$*, such that* $P_a|_{nr} = P_b|_{nr}$*, then* $\mathcal{O}(H_a, P_a)|_{n(r+1)} = \mathcal{O}(H_b, P_b)|_{n(r+1)}$.

Proof. We only need to prove that $\mathcal{O}(H_a, P_a)[k] = \mathcal{O}(H_b, P_b)[k]$ for all $0 \leq k \leq r$.

We find the smallest c such that $H_c = H_a = H_b$, $P_c|_{nk} = P_a|_{nk} = P_b|_{nk}$. Since a and b all satisfy this condition, we have $c \leq a$ and $c \leq b$. We prove that $\mathcal{O}(H_a, P_a)[k] = \mathcal{O}(H_c, P_c)[k]$ and $\mathcal{O}(H_b, P_b)[k] = \mathcal{O}(H_c, P_c)[k]$.

By Definition 3.1, we can see that for every query (H_i, P_i), $\mathcal{O}(H_i, P_i)[k]$ is either chosen uniformly randomly or there is a $j < i$ such that $\mathcal{O}(H_i, P_i)[k] = \mathcal{O}(H_j, P_j)[k]$. In the latter case, $H_j = H_i$ and $P_j|_{nk} = P_i|_{nk}$. So for (H_a, P_a), we find a sequence $(H_{a_0}, P_{a_0}) = (H_a, P_a), (H_{a_1}, P_{a_1}), ..., (H_{a_m}, P_{a_m})$, such that $a_{i+1} < a_i, O(H_{a_i}, P_{a_i})[k] = O(H_{a_{i+1}}, P_{a_{i+1}})[k]$ for $i = 0, ..., m-1$, and $\mathcal{O}(H_{a_m}, P_{a_m})[k]$ is uniformly randomly chosen. We show that $a_m = c$. We already know that $H_{a_{i+1}} = H_{a_i}$ and $P_{a_{i+1}}|_{nk} = P_{a_i}|_{nk}$ for $i = 0, ..., m-1$, so we have $H_{a_m} = H_a$, $P_{a_m}|_{nk} = P_a|_{nk}$. Then $c \leq a_m$. If $c < a_m$, by condition 2 in Definition 3.1, the first $n(k+1)$ bits of $\mathcal{O}(H_{a_m}, P_{a_m})$ cannot be randomly chosen, which makes a contradiction. Thus $c = a_m$.

So we have $\mathcal{O}(H_a, P_a)[k] = \mathcal{O}(H_c, P_c)[k]$. Similarly, there is also $\mathcal{O}(H_b, P_b)[k] = \mathcal{O}(H_c, P_c)[k]$, so $\mathcal{O}(H_a, P_a)[k] = \mathcal{O}(H_b, P_b)[k]$ for all $0 \leq k \leq r$, which means that $\mathcal{O}(H_a, P_a)|_{n(r+1)} = \mathcal{O}(H_b, P_b)|_{n(r+1)}$. □

We will now show that how we can use this online property.

Definition 3.2. *Let $f, g : (\{0,1\}^n)^+ \times (\{0,1\}^n)^+ \to \{0,1\}^n$. We define $O_{f,g}(H, P) = f(H, \varepsilon) \| f(H, P|_n) \| f(H, P|_{2n}) \| ... \| f(H, P|_{p-n}) \| g(H, P)$, $p = |P|/n$.*

Theorem 3.2. *If f, g are random oracles, then $O_{f,g}$ perfectly simulates a JAMBU-like online oracle.*

Proof. We discuss the output of $O_{f,g}$ when f, g are random oracles by the three cases in Definition 3.1. When (H_i, P_i) is called, if there is a $j < i$ with $(H_j, P_j) = (H_i, P_i)$, then $f(H_i, P_i|_{nk}) = f(H_j, P_j|_{nk})$ for $k = 0, ..., p_i - 1$, $p_i = |P_i|/n$ and $g(H_i, P_i) = g(H_j, P_j)$. So the return value satisfies condition 1 in Definition 3.1. If there are j and r satisfy condition 2, then $f(H_i, P_i|_{nk}) = f(H_j, P_j|_{nk})$ for $k = 0, ..., r$, so that the first $n(r+1)$ bits of $O_{f,g}(H_i, P_i)$ and $O_{f,g}(H_j, P_j)$ are the same. And since $f(H_i, P_i|_{nk})$, $k = r+1, ..., p_i - 1$ and $g(H_i, P_i)$ have not been queried, they return a total $n(p_i - r)$ random bits. For condition 3, we have $f(H_i, P_i|_{nk})$, $k = 0, ..., p_i - 1$ and $g(H_i, P_i)$ have not been queried, so they return a total $n(p_i + 1)$ random bits. □

4 The Security Analysis of JAMBU under Nonce Respecting Case

4.1 Security Definition

Definition 4.1. *Let $E_{\mathcal{K}} = \{E_K | K \in \mathcal{K}\}$ be the block cipher used by JAMBU. We define $\mathcal{E}[E_K](IV \| AD, P) = C \| T, E_K \in E_{\mathcal{K}}$ be the encryption procedure of JAMBU, and $\mathcal{D}[E_K](IV \| AD, C \| T) = P$ or $\perp, E_K \in E_{\mathcal{K}}$ be the decryption procedure of JAMBU described in Section 2. Then, the advantage for an adversary A in breaking JAMBU under nonce respecting case is defined as:*

$$Adv^R_{JAMBU[E_{\mathcal{K}}]}(A) = |Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}[E_K](.,.),\mathcal{D}[E_K](.,.)} \Rightarrow 1] - Pr[A^{\$(.,.),\perp(.,.)} \Rightarrow 1]|$$

where $\$(H, P)$ is a random oracle with output length $|P| + n$, and $\perp(H, CT)$ is an oracle that always returns $\perp$.

For the adversary A, we call $\mathcal{E}[E_K]$ or \$ oracle calls by E-oracle calls, and $\mathcal{D}[E_K]$ or $\perp$ oracle calls by D-oracle calls.

In the nonce respecting case, we assume that the adversary never ask two queries with a same nonce to E-oracle. Also, without loss of generalization, we suppose that the adversary should not use the result from a query of E-oracle to query D-oracle, and vice versa. Let the set of all adversaries be $\mathcal{A}$, then we define $Adv^R_{JAMBU[E]} = \max_{A\in\mathcal{A}} Adv^R_{JAMBU[E]}(A)$ by the maximal advantage over all adversaries.

If we suppose that the adversary always choose a nonce randomly as long as it has not been reused in the E-oracle queries, and call the set of all such adversaries $\mathcal{A}^\$$, we can define $Adv^{R\$}_{JAMBU[E]} = \max_{A\in\mathcal{A}^\$} Adv^R_{JAMBU[E]}(A)$.

The differences between adversaries who choose nonce randomly and arbitrarily has already been discussed by Rogaway [18]. In this paper, we will also see in our proofs that there are differences between the two models.

We also give out the PRNG version of security definition for JAMBU:

Definition 4.2. *Let $E_\mathcal{K} = \{E_K | K \in \mathcal{K}\}$ be the block cipher used by JAMBU. We define $\mathcal{G}[E_K](IV\|AD, P) = P \oplus C\|T, E_K \in E_\mathcal{K}$ be a JAMBU-based pseudorandom number generator, where IV, AD, P and C, T are inputs and outputs of the JAMBU encryption scheme. Then, the advantage for an adversary A in breaking the PRNG mode of JAMBU under the nonce respecting case is defined as:*

$$Adv^R_{JAMBU(G)[E_\mathcal{K}]}(A) = |Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{G}[E_K](.,.)} \Rightarrow 1] - Pr[A^{\$(.,.)} \Rightarrow 1]|$$

To distinguish the PRNG version from the encryption/decryption version, we call $\mathcal{G}[e]$ or \$ oracle calls in this definition by G-oracle calls.

In this definition, we only restrict the adversary A by never repeating the same nonce.

Also, we define $Adv^R_{JAMBU(G)[E]} = \max_A Adv^R_{JAMBU(G)[E]}(A)$ by the maximal advantage over all adversaries, and $Adv^{R\$}_{JAMBU(G)[E]}$ by the maximal advantage over all adversaries which choose a nonce IV randomly providing non-repeatedness.

We prove the relationship between the standard version and the PRNG version of security definition.

Theorem 4.1. *$Adv^R_{JAMBU} \leq Adv^R_{JAMBU(G)} + q/2^n$ and $Adv^{R\$}_{JAMBU} \leq Adv^{R\$}_{JAMBU(G)} + q/2^n$, q is the number of queries.*

Proof. To prove the result, we show that for any adversary A that $Adv^R_{JAMBU[E_K]}(A) = \epsilon$, we can construct an adversary A' that $Adv^R_{JAMBU(G)[E_K]}(A') \geq \epsilon - q/2^n$. This is done by simulating the E-oracle and D-oracle calls using the G-oracle.

First, we give an alternative security definition of JAMBU(G):

$$Adv^R_{JAMBU(G)'[E_K]}(A) = |Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{G}[E_K](.,.)} \Rightarrow 1] - Pr[A^{\mathcal{O}(.,.)} \Rightarrow 1]|$$

where $\mathcal{O}$ is a JAMBU-based online oracle.

By Theorem 3.2, $\mathcal{O}$ can be perfectly simulated by $O_{f,g}$, where f, g are random oracles. For $\mathcal{G}[E_K]$, by the definition of JAMBU scheme, we can see that $P[i] \oplus C[i]$ is only related to $H = IV\|AD$ and $P|_{ni} = P[0]\|...\|P[i-1]$, so for JAMBU(G), we can also define $s(H, P|_{ni}) = P[i] \oplus C[i]$ and $t(H, P) = T$ (which is the tag). So $\mathcal{G}(H, P) = s(H, P|_0)\|s(H, P|_n)\|...\|s(H, P|_{n(p-1)})\|t(H, P)$, $p = |P|/n$.

The adversary A' is obtained by (G is $\mathcal{G}[E_K]$ or $\mathcal{O}$):

(1) Replace any E-oracle calls $E(H, P)$ in A by the following $E^G(H, P)$:

Call $G(H, P)$ and return $G(H, P) \oplus (P\|0^n)$.

(2) Replace any D-oracle calls $D(H, CT)$ in A by the following $D^G(H, CT)$:

Let $C = C[0]\|...\|C[p-1]\|T$, each $C[i]$ and T is an n-bit block. First set $P \leftarrow \varepsilon$. For $i = 0, 1, ..., p-1$, let $P \leftarrow P\|(f(H, P) \oplus C[i])$ (or $P \leftarrow P\|(s(H, P) \oplus C[i])$ respectively). Finally, let $T' = g(H, P)$ ($T' = t(H, P)$ respectively). If $T' = T$, return P, else return $\perp$.

If A is called with $\mathcal{E}[E_K], \mathcal{D}[E_K]$ and A' is called with $E^{\mathcal{G}[E_K]}, D^{\mathcal{G}[E_K]}$, by the definition of JAMBU scheme, $E^{\mathcal{G}}(H, P)$ and $D^{\mathcal{G}}(H, P)$ simulate $\mathcal{E}(H, P)$ and $\mathcal{D}(H, P)$ perfectly. So $Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{G}[E_K](.,.)} \Rightarrow 1] = Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}[E_K](.,.),\mathcal{D}[E_K](.,.)} \Rightarrow 1]$.

If A is called with $\perp$ and A' is called with $D^{\mathcal{O}}$, then $D^{\mathcal{O}}$ does not return $\perp$ only when $g(H, P) = T$. Since g is a random oracle, for each query the probability is only $1/2^n$, and the total probability is no more than $q/2^n$.

If A is called with \$, A' is called with $E^{\mathcal{O}}$, and $D^{\mathcal{O}}$ always returns $\perp$, then $\mathcal{O}$ acts the same as \$. This is because when the encryption queries are nonce respecting and the decryption queries return nothing about $\mathcal{O}$, $\mathcal{O}$ never returns two queries with the same header H to the adversary A'. So $E^{\mathcal{O}}(H, P) = \mathcal{O}(H, P) \oplus (P\|0^n)$ is also a uniformly random string, hence indistinguishable from $\$(H, P)$. So $|Pr[A^{\$(.,.),\perp(.,.)} \Rightarrow 1] - Pr[A^{\mathcal{O}(.,.)} \Rightarrow 1]| \leq q/2^n$. But since $\mathcal{O}$ acts the same as \$ in this case, we also have $Pr[A^{\mathcal{O}(.,.)} \Rightarrow 1] = Pr[A^{\$(.,.)} \Rightarrow 1]$ under the condition that $D^{\mathcal{O}}$ always returns $\perp$. So $|Pr[A^{\$(.,.),\perp(.,.)} \Rightarrow 1] - Pr[A^{\$(.,.)} \Rightarrow 1]| \leq q/2^n$.

Now we have $|Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}[E_K](.,.),\mathcal{D}[E_K](.,.)} \Rightarrow 1] - Pr[A^{\$(.,.),\perp(.,.)} \Rightarrow 1]| - q/2^n \leq |Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{G}[E_K](.,.)} \Rightarrow 1] - Pr[A^{\$(.,.)} \Rightarrow 1]| \leq |Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}[E_K](.,.),\mathcal{D}[E_K](.,.)} \Rightarrow 1] - Pr[A^{\$(.,.),\perp(.,.)} \Rightarrow 1]| + q/2^n$, that is $Adv^R_{JAMBU(G)[E_K]}(A') \geq \epsilon - q/2^n$.

And that means the probability for A successfully attacks JAMBU is not greater than $Adv^R_{JAMBU(G)}(A') + q/2^n$. If $A \in \mathcal{A}$ is an adversary that can only choose nonce randomly, then A' also can only choose nonce randomly. So we have $Adv^R_{JAMBU}(A) \leq Adv^R_{JAMBU(G)} + q/2^n$ for any adversary $A \in \mathcal{A}$, which is $Adv^R_{JAMBU} \leq Adv^R_{JAMBU(G)} + q/2^n$. We also have $Adv^{R\$}_{JAMBU} \leq Adv^{R\$}_{JAMBU(G)} + q/2^n$ when we let the adversary A be taken from $\mathcal{A}^{\$}$. □

4.2 Security Proof

Now we discuss the PRNG security of JAMBU. First of all, we suppose that the adversary made totally q queries. Let those queries be (H_1, P_1),...,(H_q, P_q), $|H_j| = n(h_j+1)$, $|P_j| = np_j$, and the total length of H: $|H_1|+...+|H_q| = n(h+q)$, total length of P: $|P_1| + ... + |P_q| = np$. That means totally h blocks of ADs, totally p blocks of plaintexts.

Theorem 4.2. *Let* $Adv^R_{JAMBU(G)[Perm(2n)]}(A)$ *be defined as:*

$$Adv^R_{JAMBU(G)[Perm(2n)]}(A) = |Pr[p \xleftarrow{\$} Perm(2n) : A^{\mathcal{G}[p](.,.)} \Rightarrow 1] - Pr[A^{\$(.,.)} \Rightarrow 1]|$$

where $Perm(2n)$ *is the set of all* $2n$*-bit permutations, and* $\mathcal{G}[p]$ *is obtained by replacing* E_K *in* $\mathcal{G}[E_K]$ *by* p*. Let* $Adv_{E_K}(q)$ *be the advantage for distinguishing* $E_K \xleftarrow{\$} E_{\mathcal{K}}$ *with* $p \xleftarrow{\$} Perm(2n)$ *making at most* q *queries. Then:*

$$Adv^R_{JAMBU(G)[E_{\mathcal{K}}]} \leq Adv^R_{JAMBU(G)[Perm(2n)]} + Adv_{E_{\mathcal{K}}}(h + p + 3q).$$

Proof. We prove that the distinguishing probability between $JAMBU(G)[E_K], K \xleftarrow{\$} \mathcal{K}$ and $JAMBU(G)[p], p \xleftarrow{\$} Perm(2n)$ is not greater than $Adv_{E_{\mathcal{K}}}(h + p + 3q)$. Because the only black box in $JAMBU(G)[E_K]$ or $JAMBU(G)[p]$ is E_K or p, so for an adversary A making oracle calls to $JAMBU(G)[E_K]$ or $JAMBU(G)[p]$, we can construct an adversary A' returning the same result and making oracle calls only to E_K or p. So, the probability for A to distinguish between $JAMBU(G)[E_K]$ and $JAMBU(G)[p]$ is exactly the same with the probability for A' to distinguish between E_K and $Perm(2n)$, while A' making totally $h + p + 3q$ calls to E_K or p, by the construction of mode JAMBU.

Since A' cannot distinguish E_K with p in $h + p + 3q$ queries with probability less than $Adv_{E_{\mathcal{K}}}(h + p + 3q)$, then adversary A also cannot distinguish $JAMBU(G)[E_K]$ with $JAMBU(G)[p]$ at the same probability, which holds for any adversary A. □

Theorem 4.3. *Let* $Adv^R_{JAMBU(G)[Perm(2n)]}(A)$ *be defined as:*

$$Adv^R_{JAMBU(G)[Rand(2n)]}(A) = |Pr[f \xleftarrow{\$} Rand(2n) : A^{\mathcal{G}[f](.,.)} \Rightarrow 1] - Pr[A^{\$(.,.)} \Rightarrow 1]|$$

where $Rand(2n)$ *is the set of all functions of* $2n$ *bit input and* $2n$ *bit output, and* $\mathcal{G}[f]$ *is obtained by replacing* E_K *in* $\mathcal{G}[E_K]$ *by* f*. Then:*

$$Adv^R_{JAMBU(G)[Perm(2n)]} \leq Adv^R_{JAMBU(G)[Rand(2n)]} + (h + p + 3q)^2/2^{2n+1}.$$

Proof. See the PRP-PRF transforming lemma[5]. □

Now to the main part of the proof. We define a game which simulates $JAMBU(G)[f \xleftarrow{\$} Rand(2n)]$ queries, called JAMBU-S, and then calculate the advantage for an adversary to distinguish between it and a random oracle. Suppose that $JAMBU(G)[f]$ is queried q times, which inputs are $(H_1, P_1), ..., (H_q, P_q)$.

According to the JAMBU scheme, we divide each (H_j, P_j) into n-bit blocks, where $H_j = IV_j \| AD_j[0] \| ... \| Ad_j[h_j - 1]$, $P_j = P_j[0] \| ... \| P_j[p_j - 1]$, $h_j = |H_j|/n - 1$ and $p_j = |P_j|/n$. The game is defined as (here we use $\perp(.)$ for a function that is undefined everywhere):

Game 1 JAMBU-S (and JAMBU-R by eliminating the shadowed part)

$S \leftarrow \emptyset$; $\pi \leftarrow \perp(.)$; $bad \leftarrow false$.
for $j = 1$ **to** q **do**
  $U \leftarrow 0^n$; $V \leftarrow IV_j$; $R \leftarrow 0^n$;
  if $U\|V \in S$ **then** $bad \leftarrow true$; $X\|Y \leftarrow \pi(U\|V)$; **else**
    $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; // Initialization Stage
  $U \leftarrow X$; $V \leftarrow Y \oplus 5$; $R \leftarrow R \oplus U$;
  if $U\|V \in S$ **then** $bad \leftarrow true$; $X\|Y \leftarrow \pi(U\|V)$; **else**
    $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; // IV processing Stage
  for $i = 0$ **to** $h_j - 1$ **do**
    $U \leftarrow X \oplus AD_j[i]$; $V \leftarrow Y \oplus R \oplus 1$; $R \leftarrow R \oplus U$;
    if $U\|V \in S$ **then** $bad \leftarrow true$; $X\|Y \leftarrow \pi(U\|V)$; **else**
      $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; // AD processing Stage
  end for
  for $i = 0$ **to** $p_j - 1$ **do**
    $U \leftarrow X \oplus P_j[i]$; $V \leftarrow Y \oplus R \oplus 1$; $R \leftarrow R \oplus U$; **output** V;
    if $U\|V \in S$ **then** $bad \leftarrow true$; $X\|Y \leftarrow \pi(U\|V)$; **else**
      $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; // Plaintext processing Stage
  end for
  $U \leftarrow X$; $V \leftarrow Y \oplus 3$; $R \leftarrow R \oplus U$;
  if $U\|V \in S$ **then** $bad \leftarrow true$; $X\|Y \leftarrow \pi(U\|V)$; **else**
    $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; // Finalization Stage
  output $X \oplus Y \oplus R$;
end for

We divide the game into five stages: initialization stage; IV processing stage, AD processing stage, plaintext processing stage, and finalization stage. Note that for the convenience of further discussion, the stages we divided here is slightly different from the JAMBU document as what we introduced in Section 2.

The states of JAMBU-S are U, V, R, X, Y. Since the states are dynamic, we add a subscript t called *time*, and write them as U_t, V_t, R_t, X_t, Y_t. $t \leftarrow 0$ when the game starts, and $t \leftarrow t + 1$ each time after $X\|Y$ is updated. That is, when time

is set to t, first update update $U_{t-1}\|V_{t-1}$ into $U_t\|V_t$ using $X_{t-1}\|Y_{t-1}$ and R_{t-1}, then update R_{t-1} into R_t using U_t, and finally update $X_{t-1}\|Y_{t-1}$ into $X_t\|Y_t$ using $U_t\|V_t$. We also write S_t, bad_t be the value of S after $X_t\|Y_t$ is updated. It is easy to see that t is at one of the five stages. We can also drop the t if there is no confusion.

We also define a simplified version for simulator JAMBU-S, called JAMBU-R, by eliminating the shadowed part of JAMBU-S. (Here R means the nonce respecting case.)

We can see that JAMBU-S and JAMBU-R act differently only when $bad = true$ in JAMBU-S.

Theorem 4.4. *In the JAMBU-S simulator, let $U_t\|V_t$ be the current state. Then if t is not at the initialization stage and $bad_{t-1} = false$, $U_t\|V_t$ is uniformly random and independent with elements in S_{t-1}.*

Proof. If t is not at the initialization stage, then one of the four cases holds, depends on which stage t is at:

(1) $U_t = X_{t-1}, V_t = Y_{t-1} \oplus 5$;
(2) $U_t = X_{t-1} \oplus AD_j[i], V_t = Y_{t-1} \oplus R_{t-1} \oplus 1$ for some i, j;
(3) $U_t = X_{t-1} \oplus P_j[i], V_t = Y_{t-1} \oplus R_{t-1}$ for some i, j;
(4) $U_t = X_{t-1}, V_t = Y_{t-1} \oplus 3$.

Since $bad = false$, we have $X_{t-1}\|Y_{t-1} \xleftarrow{\$} \{0,1\}^{2n}$, so $X_{t-1}\|Y_{t-1}$ is uniformly random and independent with all previous states and outputs, including elements in S_{t-1}. So for case (1) or case (4), $U_t\|V_t = X_{t-1}\|Y_{t-1} \oplus 5$ or $X_{t-1}\|Y_{t-1} \oplus 3$ is uniformly random and independent with elements in S_{t-1}. For case (2) or case (3), we have $U_t\|V_t = (X_{t-1}\|Y_{t-1}) \oplus (AD_j[i]\|R_{t-1} \oplus 1)$ or $U_t\|V_t = (X_{t-1}\|Y_{t-1}) \oplus (P_j[i]\|R_{t-1})$. The adversary can only choose $AD_j[i]$ or $P_j[i]$ according to the previous outputs of JAMBU-S, which are independent with $X_{t-1}\|Y_{t-1}$, so $AD_j[i]$ or $P_j[i]$ is also independent with $X_{t-1}\|Y_{t-1}$. Also, $X_{t-1}\|Y_{t-1}$ is independent with all previous states, including elements in S_{t-1} and R_{t-1}. Then we have $U_t\|V_t$ is uniformly random and independent with elements in S_{t-1}. □

Now for our main theorem:

Theorem 4.5. $Adv^{R\$}_{JAMBU(G)[Rand(2n)]} \leq (h + p + 3q)^2/2^{2n+1}$.

Proof. Compared to the construction of JAMBU, we can see that JAMBU-S acts the same as $JAMBU(G)[f \xleftarrow{\$} Rand(2n)]$, and JAMBU-R always returns a random string. So $Adv_{JAMBU(G)[Rand(2n)]}(A) = |Pr[f \xleftarrow{\$} Rand(2n) : A^{\mathcal{G}[f](.,.)} \Rightarrow 1] - Pr[A^{\$(.,.)} \Rightarrow 1]|$ is no more than the probability that JAMBU-S and JAMBU-R act differently, that is $bad = true$ in JAMBU-S.

Now we calculate the probability that $bad \leftarrow true$. If $bad = false$ at time $t-1$, then the probability of $bad = true$ at time t is that $U_t\|V_t \in S_{t-1}$. We discuss the value $U_t\|V_t$ in the following cases:

(1) t is not at the initialization stage. Then by Theorem 4.2, $U_t\|V_t$ is an uniformly random $2n$-bit string and independent with elements in S_{t-1}. Then $Pr(U_t\|V_t \in S_{t-1}) = |S_{t-1}|/2^{2n}$.

(2) t is at the initialization stage, and $V_t = IV_j$ for some j. Then for each $s \in S$, suppose that $s = U_s \| V_s$. If s is also the initialization stage, since the adversary never repeat a nonce, $Pr(U_t \| V_t = U_s \| V_s) = 0$. Otherwise, $U_s \| V_s$ is uniformly random. So $Pr(U_s = 0^n) = 1/2^n$. Since we suppose that IV_j is chosen randomly, then $Pr(V_s = IV_j) = 1/2^n$. Also U_s and V_s are independent, so $Pr(U_t \| V_t = U_s \| V_s) = 1/2^n$. Then we have $Pr(U_t \| V_t \in S_t) \leq |S_t|/2^{2n}$.

Also, we have that for each query $(IV_i \| AD_i, P_i)$, the states are updated $h_i + p_i + 3$ times, the total states are updated $h+p+3q$ times. So $|S| \leq h+p+3q$. Then the total probability $Pr(bad = true) \leq 1/2^{2n} + 2/2^{2n} + ...(h + p + 3q - 1)/2^2 n \leq (h + p + 3q)^2/2^{2n+1}$. □

Here we have the conclusion:

Theorem 4.6. $Adv^{R\$}_{JAMBU(G)[E_K]} \leq Adv_{E_K}(h + p + 3q) + (h + p + 3q)^2/2^{2n}$; $Adv^{R\$}_{JAMBU[E_K]} \leq Adv_{E_K}(h + p + 3q) + (h + p + 3q)^2/2^{2n} + q/2^n$.

Proof. Add up all the probabilities. □

Note that above we suppose that the adversary can only choose a nonce randomly, which is usually not the case. So we need a further discussion on the distinguish probability where the adversary can choose a nonce according to the previous output of queries, which we defined the probability as $Adv^{R}_{JAMBU(G)[E_K]}$. We have the following result:

Theorem 4.7. *Let $r_1, ..., r_y \in \{1, 2, ..., x\}$ be uniformly random variables, and $c_i = |\{r_j | r_j = i\}|$ be the number of variables taken value i. $M(x, y) = \max_{1 \leq i \leq x} c_i$, and $EM(x, y)$ be the mathematical expectation of $M(x, y)$, the maximal value among c_i. Then, we have $Adv^{R}_{JAMBU[E_K]} \leq Adv_{E_K}(h+p+3q)+(h+p+3q)^2/2^{2n}+ q/2^n + 2^{-n}\Sigma_{1 \leq i \leq q} EM(2^n, \Sigma_{1 \leq j \leq i-1}(h_j + p_j + 3))$.*

Proof. Similarly, we compute the probability of $bad = true$. It is the same as in the proof of Theorem 4.3 that when t is not at the initialization stage, $Pr(U_t \| V_t \in S_{t-1}) = |S_{t-1}|/2^{2n}$.

Now we suppose that t is at the initialization stage, and $V_t = IV_j$ for some j. For any $U_s \| V_s \in S_{t-1}$, if s is the initialization stage, then $U_s \| V_s \neq U_t \| V_t$ since the adversary never repeats a nonce. Otherwise, the probability is $Pr(U_s = 0^n) Pr(V_s = IV_j) = Pr(V_s = IV_j)/2^n$. Since IV_j is chosen by the adversary, then the adversary can choose a value that $|\{s | s \in S_{t-1} \wedge s|_{n,2n} = IV_j\}|$ takes the maximum value, which is what we defined as $M(2^n, |S_{t-1}|)$. So in the i-th query, we have the probability $EM(2^n, \Sigma_{1 \leq j \leq i-1}(h_j + p_j + 3))/2^n$.

When we sum up all the probabilities, we have $Adv^{R}_{JAMBU[E_K]} \leq Adv_{E_K}(h+ p+3q)+(h+p+3q)^2/2^{2n}+q/2^n+2^{-n}\Sigma_{1 \leq i \leq q} EM(2^n, \Sigma_{1 \leq j \leq i-1}(h_j+p_j+3))$. □

The exact value for $EM(x, y)$ is extremely hard to calculate. But it has been proven in [11, 13], if $y/x = c$ is a constant, then $\lim_{x \to \infty} EM(x, y) = \lceil \log x / \log \log x \rceil$. So we suggest that $EM(2^n, \Sigma_{1 \leq j \leq i}(h_j + p_j + 3)) = O(n/\log n))$, and thus $Adv^{R}_{JAMBU[E_K]} \leq Adv_{E_K}(h + p + 3q) + (h + p + 3q)^2/2^{2n} + q/2^n +$

$O(qn/(2^n \log n))$, which is slightly lower than the birthday bound. Although the effect is very small, but it should still be taken into account if there is a strict demand on security.

5 The Security Analysis of JAMBU under Nonce Misuse Case

The security notion of JAMBU under nonce misuse case is harder to define. If two inputs $(IV\|AD, P)$ and $(IV\|AD, P')$ share the same header and the first i plaintext blocks, according to the encryption scheme, their $i+1$-th ciphertext block $C[i+1]$ and $C'[i+1]$ have the relationship $C[i+1] \oplus C'[i+1] = P[i+1] \oplus P'[i+1]$ and are insecure. This is because the first $i+1$ blocks of PRNG output is only related to the header $IV\|AD$ and the first i plaintext blocks. So rather than the security of encryption/decryption mode of JAMBU, it is better that we use JAMBU-based PRNG to define the security.

Definition 5.1. *Let $E_{\mathcal{K}} = \{E_K | K \in \mathcal{K}\}$ be the block cipher used by JAMBU, and $\mathcal{G}[E_K](IV||AD, P) = P \oplus C\|T$ be the JAMBU-based PRNG. The advantage for an adversary A in breaking the PRNG mod of JAMBU under nonce misuse case is defined by:*

$$Adv^M_{JAMBU(G)[E_{\mathcal{K}}]}(A) = |Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{G}[E_K](.,.)} \Rightarrow 1] - Pr[A^{\mathcal{O}(.,.)} \Rightarrow 1]|$$

Where $\mathcal{O}$ is a JAMBU-like online oracle.

Also, we define $Adv^M_{JAMBU(G)[E_{\mathcal{K}}]} = \max_A Adv^M_{JAMBU(G)[E_{\mathcal{K}}]}(A)$ by the maximal advantage over all adversaries.

Theorem 5.1. $Adv^M_{JAMBU(G)[Rand(2n)]} \leq (h+p+3q)^2/2^{n+1}$.

The complete proof of this theorem is given in Appendix B.

Proof Sketch. We use the JAMBU-like online oracle defined in Section 3 for the ideal security of JAMBU-based PRNG under nonce misuse case. Like the game JAMBU-R, we define a game JAMBU-M that perfectly simulates the JAMBU-like online oracle, so the distinguishing probability turns into the probability that JAMBU-S and JAMBU-M act differently. The *bad* event defined in Section 4 could be divided into two parts. One is that the current query part which has been input into the game is a prefix of some previous query. We write this event by bad'. In this case, JAMBU-S has the same states and outputs of that previous query, which acts the same as a JAMBU-like online oracle or JAMBU-M. The other case is that bad' does not occur, so there is no previous query which the current query part is its prefix, while the current call to f still collides with some previous call. This is the real distinguishing probability, and we call the event bad^*, bad^* occurs when $bad = ture$ and $bad' = false$. We show that this collision probability for two inputs of f, say $U_t\|V_t = U_s\|V_s$ is not greater than $1/2^n$ (instead of $1/2^{2n}$ in the nonce respecting case).

We discuss the probability by three cases: (1) t or s is at initialization stage; (2) t and s are the first block after a common prefix; (3) other case. For case (1), like in nonce respecting case, it is easily shown that $Pr(U_t\|V_t = U_s\|V_s) \leq 1/2^n$. For case (2), either the current input blocks are different at s, t, or the stages are different at s, t. Since $t-1$ and $s-1$ correspond to the same prefix, it can be shown that $X_{t-1}\|Y_{t-1} = X_{s-1}\|Y_{s-1}$. By the JAMBU scheme, if the input blocks are different, $U_t \neq U_s$, if the stages are different, $V_t \neq V_s$. For case (3), we prove that U_t and U_s are uniformly random and independent, so $Pr(U_t = U_s) = 1/2^n$. Then, we have the total probability is no more than $1/2^n + ... + (h+p+3q-1)/2^n \leq (h+p+3q)^2/2^{n+1}$.

Theorem 5.2. $Adv^M_{JAMBU(G)[E_\mathcal{K}]} \leq Adv_{E_K}(h+p+3q) + (h+p+3q)^2/2^{2n+1} + (h+p+3q)^2/2^{n+1}$.

Proof. Like nonce respecting case, we have $Adv^M_{JAMBU(G)[E_\mathcal{K}]} - Adv^M_{JAMBU(G)[Perm(2n)]} \leq Adv_{E_K}(h+p+3q)$ and $Adv^M_{JAMBU(G)[Perm(2n)]} - Adv^M_{JAMBU(G)[Rand(2n)]} \leq (h+p+3q)^2/2^{2n+1}$. We omit the details. Also we proved that $Adv^M_{JAMBU(G)[Rand(2n)]} \leq (h+p+3q)^2/2^{n+1}$, add up the three inequations to get the result. □

6 Conclusion

In this paper, we discussed the provable security of JAMBU in both nonce respecting and nonce misuse cases. Since the designers did not give the security proof in their submission, we believe that our work is an important complement, especially in the nonce misuse case, where the original security claim of designers has been overthrown by other researchers.

Also our proofs themselves are quite technical, and shared some insights on the potential users of JAMBU. In the nonce respecting case, we shown that when the adversary can choose a nonce (which is a quite reasonable assumption), the security cannot achieve the birthday bound, although very close. If the underlying cipher of JAMBU is a lightweight block cipher where the birthday bound could be reached, this small effect on security must be taken into account by users. In the nonce misuse case, we show that the security is $n/2$ bits, and since there is an attack with $O(2^{n/2})$ queries, this security bound is tight. Our proof can be a guidance for application.

In their submission, the designers claimed that the confidentiality is the same as the key length. However, by using provable security method, we can only prove a security up to its birthday bound, which is n-bit. (There are some modes of operation which have a security beyond the birthday bound, but JAMBU is clearly not designed for a beyond-birthday-bound security.) This does not necessarily mean that there exists an attack which is higher than n but lower than the key length under the nonce respecting case. Also, we use a single security claim to capture both confidentiality and integrity, which is different from the security claim by the designers, since we use a different security model.

We hope that, by giving security proofs on JAMBU under both nonce respecting and nonce misuse case, we can help strengthening the competitiveness of JAMBU, and bring the scheme further to practical use.

References

1. Andreeva E, Bogdanov A, Luykx A, et al. Parallelizable and authenticated online ciphers. International Conference on the Theory and Application of Cryptology and Information Security. Springer Berlin Heidelberg, 2013: 424-443.
2. Andreeva E, Bogdanov A, Datta N, et al. COLM v1. CAESAR competition proposal, 2016.
3. Bellare M, Namprempre C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. International Conference on the Theory and Application of Cryptology and Information Security. Springer Berlin Heidelberg, 2000: 531-545.
4. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976. Springer, 2000: 317-330.
5. Bellare M, Rogaway P. Code-Based Game-Playing Proofs and the Security of Triple Encryption. IACR Cryptology ePrint Archive, 2004, 2004: 331.
6. Fleischmann E, Forler C, Lucks S. McOE: a family of almost foolproof on-line authenticated encryption schemes. Fast Software Encryption. Springer Berlin Heidelberg, 2012: 196-215.
7. Gueron S, Lindell Y. GCM-SIV: Full nonce misuse-resistant authenticated encryption at under one cycle per byte. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015: 109-119.
8. Hoang V T, Reyhanitabar R, Rogaway P, et al. Online authenticated-encryption and its nonce-reuse misuse-resistance. Annual Cryptology Conference. Springer Berlin Heidelberg, 2015: 493-517.
9. Iwata T, Minematsu K, Guo J, et al. CLOC: Authenticated encryption for short input. International Workshop on Fast Software Encryption. Springer Berlin Heidelberg, 2014: 149-167.
10. Iwata T, Yasuda K. HBS: A single-key mode of operation for deterministic authenticated encryption. Fast Software Encryption. Springer Berlin Heidelberg, 2009: 394-415.
11. Kimber A C. A note on Poisson maxima. Zeitschrift fr Wahrscheinlichkeitstheorie und Verwandte Gebiete, 1983, 63(4): 551-552.
12. Minematsu K. AES-OTR v3. CAESAR competion proposal, 2016.
13. Mitzenmacher M. The power of two choices in randomized load balancing. IEEE Transactions on Parallel and Distributed Systems, 2001, 12(10): 1094-1104.
14. McGrew D, Viega J. The Galois/counter mode of operation (GCM). Submission to NIST. http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf, 2004.
15. McGrew D A, Viega J. The security and performance of the Galois/Counter Mode (GCM) of operation. International Conference on Cryptology in India. Springer Berlin Heidelberg, 2004: 343-355.
16. Peyrin T, Sim S M, Wang L, et al. Cryptanalysis of JAMBU. International Workshop on Fast Software Encryption. Springer Berlin Heidelberg, 2015: 264-281.

17. Rogaway P. Authenticated-encryption with associated-data. Proceedings of the 9th ACM conference on Computer and communications security. ACM, 2002: 98-107.
18. Rogaway P. Nonce-based symmetric encryption. International Workshop on Fast Software Encryption. Springer Berlin Heidelberg, 2004: 348-358.
19. Rogaway P. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. International Conference on the Theory and Application of Cryptology and Information Security. Springer Berlin Heidelberg, 2004: 16-31.
20. Rogaway P, Bellare M, Black J. OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Transactions on Information and System Security (TISSEC), 2003, 6(3): 365-403.
21. Rogaway P, Shrimpton T. A provable-security treatment of the key-wrap problem. Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2006: 373-390.
22. Rogaway P, Shrimpton T. Deterministic Authenticated-Encryption. Advances in Cryptology (EUROCRYPT). 2007, 6.
23. Reyhanitabar R, Vaudenay S, Vizár D. Misuse-resistant variants of the OMD authenticated encryption mode. International Conference on Provable Security. Springer International Publishing, 2014: 55-70.
24. Wu H, Huang T. JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU. CAESAR competition proposal, 2014.
25. Wu H, Huang T. The JAMBU Lightweight Authentication Encryption Mode (v2.1). CAESAR competition proposal, 2016.

A Proof of Theorem 5.1

First, we define another game called JAMBU-M to simulate the ideal security under nonce misuse case:

We note that in JAMBU-M, $\pi_3(H, \varepsilon)$ is always undefined. For the simplicity of further discussion, if (H_j, P_j) is one of the queries, we can let $\pi_3(H_j, \varepsilon) = \pi_2(H_j)$ if $\pi_2(H_j)$ has already been defined. So we always have that the i-th output block $S[i] = \pi_3(H, P|_{ni})|_{n,2n} \oplus R_t$ for some t.

For each time t, we write Tr_t for all input blocks up to the time of the current query. That is, if t is at the initial or IV processing stage, $Tr_t = IV$, if t is at the AD processing stage, $Tr_t = IV\|AD|_{n(i+1)}$ for some i, if t is at the plaintext processing stage, $Tr_t = (IV\|AD, P|_{n(i+1)})$ for some i, if t is at the finalization stage, $Tr_t = (IV\|AD, P)$. Then, the conditional statement of $\pi_1, ..., \pi_4$ can be unified into: if $\pi_i(Tr_t)$ is defined, then $X_t\|Y_t = \pi_i(Tr_t)$, else $X_t\|Y_t = \pi_i(Tr_t) \xleftarrow{\$} \{0,1\}^{2n}$, $i = 1, ..., 5$. We can also use the same notion Tr_t for the game JAMBU-S. We say s, t are equal if $Tr_s = Tr_t$ and s, t are at the same stage.

Lemma A.1. *If s, t are equal, then $U_s = U_t, V_s = V_t, R_s = R_t, X_s = X_t, Y_s = Y_t$ for both JAMBU-S and JAMBU-M.*

Proof. JAMBU-S or JAMBU-M can be equivalently written as JAMBU-S$^{\pi}$ or JAMBU-M$^{\pi_1,\pi_2,\pi_3,\pi_4}$ where $\pi, \pi_1, \pi_2, \pi_3, \pi_4$ are oracles, also JAMBU-S and JAMBU-M are deterministic algorithms. So with the same input $Tr_s = Tr_t$ and the same stage, the states of JAMBU-S or JAMBU-M are always the same. □

Game 2 JAMBU-M

$\pi_1, \pi_2, \pi_3, \pi_4 \leftarrow \perp(.)$; $bad^* \leftarrow false$.
for $j = 1$ **to** q **do**
  $U \leftarrow 0^n$; $V \leftarrow IV_j$; $R \leftarrow 0^n$;
  if $\pi_1(IV_j) \neq \perp$ **then** $X\|Y \leftarrow \pi_1(IV_j)$; **else**
    $X\|Y \leftarrow \pi_1(IV) \xleftarrow{\$} \{0,1\}^{2n}$; // Initialization Stage
  $U \leftarrow X$; $V \leftarrow Y \oplus 5$; $R \leftarrow R \oplus U$;
  if $\pi_2(IV_j) \neq \perp$ **then** $X\|Y \leftarrow \pi_2(IV_j)$; **else**
    $X\|Y \leftarrow \pi_2(IV) \xleftarrow{\$} \{0,1\}^{2n}$; // IV processing Stage
  for $i = 0$ **to** $h_j - 1$ **do**
    $U \leftarrow X \oplus AD_j[i]$; $V \leftarrow Y \oplus R \oplus 1$; $R \leftarrow R \oplus U$;
    if $\pi_2(IV_j\|AD_j|_{n(i+1)}) \neq \perp$ **then** $X\|Y \leftarrow \pi_2(IV_j\|AD_j|_{n(i+1)})$; **else**
      $X\|Y \leftarrow \pi_2(IV_j\|AD_j|_{n(i+1)}) \xleftarrow{\$} \{0,1\}^{2n}$; // AD processing Stage
  end for
  for $i = 0$ **to** $p_j - 1$ **do**
    $U \leftarrow X \oplus P_j[i]$; $V \leftarrow Y \oplus R$; $R \leftarrow R \oplus U$; **output** V;
    if $\pi_3(IV_j\|AD_j, P_j|_{n(i+1)}) \neq \perp$ **then** $X\|Y \leftarrow \pi_3(IV_j\|AD_j, P_j|_{n(i+1)})$; **else**
      $X\|Y \leftarrow \pi_3(IV_j\|AD_j, P_j|_{n(i+1)}) \xleftarrow{\$} \{0,1\}^{2n}$; // Plaintext processing Stage
  end for
  $U \leftarrow X$; $V \leftarrow Y \oplus 3$; $R \leftarrow R \oplus U$;
  if $\pi_4(IV_j\|AD_j, P_j) \neq \perp$ **then** $bad \leftarrow true$; $X\|Y \leftarrow \pi_4(IV_j\|AD_j, P_j)$; **else**
    $X\|Y \leftarrow \pi_4(IV_j\|AD_j, P_j) \xleftarrow{\$} \{0,1\}^{2n}$; // Finalization Stage
  output $X \oplus Y \oplus R$;
end for

Now, we have a JAMBU-S game which simulates the JAMBU scheme, and a JAMBU-M game which acts the same as JAMBU-like online oracle, what we defined as perfect nonce misuse security. Then, what remains is to calculate the advantage for distinguishing between the two games. To get the distinguishing probability, we add an event bad^* to the simulator JAMBU-S, without changing its behaviour. The newly defined JAMBU-S' is as follows:

We also write bad_t, bad'_t, bad^*_t as the value of bad, bad', bad^* at time t. It is easy to see that $bad'_t = true$ only if there is no $s < t$ such that s, t are equal.

Lemma A.2. *(1) For any t, the state X_t is independent to all outputs in JAMBU-M.*

(2) Let $Y_s \oplus R_s$ (or $X_s \oplus Y_s \oplus R_s$) and $Y_s \oplus R_s$ (or $X_s \oplus Y_s \oplus R_s$) be two output blocks. If s, t are not equal, the two output blocks are independent.

Proof. First, we show that for any t, X_t, Y_t, R_t are pairwise independent. By Lemma 5.1, we can suppose that there is no $t' < t$ that $Tr_{t'} = Tr_t$. Then $X_t \| Y_t \xleftarrow{\$} \{0,1\}^{2n}$, so $X_t \| Y_t$ is independent to all previous states and outputs, including R_t. Also $X_t \| Y_t$ is uniformly random, so X_t and Y_t are also independent.

(1) For each output block S, we have $S = Y_s \oplus R_s$ or $S = X_s \oplus Y_s \oplus R_s$ for some s, then Y_s is uniformly random and independent to X_s, R_s and X_t, so X_t and $Y_s \oplus R_s$ or $X_s \oplus Y_s \oplus R_s$ are independent.

(2) If $Tr_s \neq Tr_t$ or s, t are from different stages, $X_s \| Y_s$ and $X_t \| Y_t$ are assigned independently, and independent to R_s or R_t. So $Y_t \oplus R_t$ or $X_t \oplus Y_t \oplus R_t$ is independent with $Y_s \oplus R_s$ or $X_s \oplus Y_s \oplus R_s$. □

Theorem A.1. *JAMBU-M perfectly simulates a JAMBU-like online oracle.*

Proof. Since a JAMBU-like online oracle returns by three cases, we study the states and outputs of JAMBU-M also by the three cases:

(1) (H, P) has been already queried. Then, in JAMBU-M, we have that $\pi_1(IV)$, $\pi_2(IV \| AD|_{ni}), i = 0, ..., h$, $\pi_3(IV \| AD, P|_{ni}), i = 1, ..., p$, $\pi_4(IV \| AD, P)$ are defined. So each time $X \| Y$ is assigned the same value as the last time (H, P) was queried. By Lemma 5.1, the states of the two queries are always the same, so the outputs are also the same.

(2) (H, P) has not been queried, but there exists a queried $(H_j, P_j) = (H, P')$ and r that satisfy condition 2 in Definition 3.1. Then, we have that $\pi_1(IV)$, $\pi_2(IV \| AD|_{ni})$, $i = 0, ..., h$, $\pi_3(IV \| AD, P|_{ni})$, $i = 1, ..., r$ are defined, but $\pi_3(IV \| AD, P|_{ni})$, $i = r+1, ..., p$, $\pi_4(IV \| AD, P)$ are not. By Lemma 5.1, until the first r blocks of P have been input, the states of (H, P) and (H, P') are still the same, which means that the first $r+1$ output blocks are the same.

Since $\pi_3(IV \| AD, P|_{ni})$, $i = r+1, ..., p$, $\pi_4(IV \| AD, P)$ are undefined, so for each of the last $p - r$ output blocks, say $Y_t \oplus R_t$ or $X_t \oplus Y_t \oplus R_t$, there is no $s < t$ that $Tr_s = Tr_t$ and s, t are from the same stage. By Lemma A.2, the output block at t is independent with all previous output blocks, so the last $n(p - r)$ output bits are indistinguishable from a random assignment. That is, JAMBU-M acts the same as JAMBU-like online oracle in this case.

Game 3 JAMBU-S'

$S \leftarrow \emptyset$; $\pi, \pi_1, \pi_2, \pi_3, \pi_4 \leftarrow \perp(.)$; $bad^* \leftarrow false$.
for $j = 1$ **to** q **do**
  $U \leftarrow 0^n$; $V \leftarrow IV_j$; $R \leftarrow 0^n$;
  if $U\|V \in S$ **then** $X\|Y \leftarrow \pi(U\|V)$; $bad \leftarrow true$; **else**
    $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; $bad \leftarrow false$;
  if $\pi_1(IV_j) = \perp$ **then** $\pi_1(IV_j) \leftarrow X\|Y$; $bad' \leftarrow false$; **else** $bad' \leftarrow true$;
  if $bad = true$ and $bad' = false$ **then** $bad^* \leftarrow true$; // Initialization Stage
  $U \leftarrow X$; $V \leftarrow Y \oplus 5$; $R \leftarrow R \oplus U$;
  if $U\|V \in S$ **then** $X\|Y \leftarrow \pi(U\|V)$; $bad \leftarrow true$; **else**
    $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; $bad \leftarrow false$;
  if $\pi_2(IV_j) = \perp$ **then** $\pi_2(IV_j) \leftarrow X\|Y$; $bad' \leftarrow false$; **else** $bad' \leftarrow true$;
  if $bad = true$ and $bad' = false$ **then** $bad^* \leftarrow true$; // IV processing Stage
  for $i = 0$ **to** $h_j - 1$ **do**
    $U \leftarrow X \oplus AD_j[i]$; $V \leftarrow Y \oplus R \oplus 1$; $R \leftarrow R \oplus U$;
    if $U\|V \in S$ **then** $X\|Y \leftarrow \pi(U\|V)$; $bad \leftarrow true$; **else**
      $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; $bad \leftarrow false$;
    if $\pi_2(IV_j\|AD_j|_{n(i+1)}) = \perp$ **then** $\pi_2(IV_j\|AD_j|_{n(i+1)}) \leftarrow X\|Y$; $bad' \leftarrow false$; **else** $bad' \leftarrow true$;
    if $bad = true$ and $bad' = false$ **then** $bad^* \leftarrow true$; // AD processing Stage
  end for
  for $i = 0$ **to** $p_j - 1$ **do**
    $U \leftarrow X \oplus P_j[i]$; $V \leftarrow Y \oplus R$; $R \leftarrow R \oplus U$; **output** V;
    if $U\|V \in S$ **then** $X\|Y \leftarrow \pi(U\|V)$; $bad \leftarrow true$; **else**
      $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; $bad \leftarrow false$;
    if $\pi_3(IV_j\|AD_j, P_j|_{n(i+1)}) = \perp$ **then** $\pi_3(IV_j\|AD_j, P_j|_{n(i+1)}) \leftarrow X\|Y$; $bad' \leftarrow false$; **else** $bad' \leftarrow true$;
    if $bad = true$ and $bad' = false$ **then** $bad^* \leftarrow true$; // Plaintext processing Stage
  end for
  $U \leftarrow X$; $V \leftarrow Y \oplus 3$; $R \leftarrow R \oplus U$;
  if $U\|V \in S$ **then** $X\|Y \leftarrow \pi(U\|V)$; $bad \leftarrow true$; **else**
    $S \leftarrow S \cup \{U\|V\}$; $X\|Y \leftarrow \pi(U\|V) \xleftarrow{\$} \{0,1\}^{2n}$; $bad \leftarrow false$;
  if $\pi_4(IV_j\|AD_j, P_j) = \perp$ **then** $\pi_4(IV_j\|AD_j, P_j) \leftarrow X\|Y$; $bad' \leftarrow false$; **else** $bad' \leftarrow true$;
  if $bad = true$ and $bad' = false$ **then** $bad^* \leftarrow true$; // Finalization Stage
  output $X \oplus Y \oplus R$;
end for

(3) There exists no P' such that (H, P') has already been queried. So $(H, P|_{ni}) \notin dom(\pi_3)$, $i = 0, ..., p$ and $(H, P) \notin dom(\pi_4)$. Like case (2), we also have the $n(p+1)$ bit outputs are indistinguishable from a random assignment, the same as the output of a JAMBU-like online oracle. □

In some cases, a JAMBU-like online oracle output block is not random. By defining the bad^* event, we excluded these cases, only focus on where the JAMBU-like online oracle output block is random, but the output of real JAMBU scheme is not. In fact, we can prove the following:

Theorem A.2. *An adversary can distinguish between JAMBU-S' and JAMBU-like online oracle only when bad^* is set to true in JAMBU-S'.*

Proof. Since JAMBU-like online oracle can be simulated by game JAMBU-M, we only need to distinguish between the two games. So what we need to prove is that when bad^* is false, JAMBU-S' acts the same as JAMBU-M simulator. We prove it by induction.

At the beginning, π and $\pi_1, ..., \pi_4$ are everywhere undefined, so there are no differences between JAMBU-S and JAMBU-M. We suppose that up onto time $t-1$, JAMBU-S still acts the same as JAMBU-S'(M), and we discuss the two games at time t. There are two cases where bad^* is false at time t, the first is that $bad_t = false$ and $bad'_t = false$. In this case, $X_t \| Y_t$ is assigned to a random $2n$-bit string both in JAMBU-S and JAMBU-M, where the two games act the same.

The other case is that $bad'_t = true$. Then $\pi_i(Tr_t)$ is already defined, which i represents the current stage. So there must be some $s < t$ that s, t are equal. By Lemma A.1, the states at time t is assigned the same as states at time s for both JAMBU-S and JAMBU-M, then the two game also act the same. □

By Theorem A.2, we only need to calculate the probability for bad^* set to true at each step α. We prove that for each time t, if $bad^*_{t-1} = false$, then the probability of $bad^*_t = true$ is not greater than $|S_{t-1}|/2^n$.

If there exists $s < t$ that s, t are equal, then $bad'_t = true$, so $bad^*_t = false$. Otherwise, for any $U_s \| V_s \in S_{t-1}$, we calculate the probability of $U_t \| V_t = U_s \| V_s$. We discuss the probability by four cases:

(1) t is at the initialization stage, so $U_t \| V_t = 0^n \| IV_j$ for some j. If s is also at the initialization stage, since $bad'_t = false$, so $\pi_1(IV)$ is undefined before time t, which means that there is no $j' < j$ that $IV_j = IV_{j'}$. So $U_t \| V_t \neq U_s \| V_s$. Otherwise, one of the following holds: $U_s = X_s$, $U_s = X_s \oplus AD_k[i]$ for some i, k, $U_s = X_s \oplus P_k[i]$ for some i, k. $AD_k[i]$ or $P_k[i]$ could be chosen by the adversary, so it can be written as a function of all previous outputs. By Lemma 5.1, X_s is independent with all outputs, hence independent with $AD_k[i]$ or $P_k[i]$. We also have that X_s is uniformly random, so U_s is also uniformly random, and $Pr(U_s = 0^n) = 1/2^n$. Then $Pr(U_t \| V_t = U_s \| V_s) \leq 1/2^n$.

(2) $s-1, t-1$ are equal, and s, t are at the same stage. We first show at time t or s, JAMBU-S must be at the AD processing or plaintext processing stage. If s, t are at the IV processing stage or finalization stage, $Tr_{t-1} = Tr_t$

and $Tr_{s-1} = Tr_s$, so $Tr_t = Tr_s$, which contradicts with $bad'_t = false$. Then, we have that $U_t = X_{t-1} \oplus AD_j[i]$ (or $X_{t-1} \oplus P_j[i]$) and $U_s = X_{s-1} \oplus AD_k[i]$ (or $X_{s-1} \oplus P_k[i]$) for some i, j, k. By Lemma 5.2, $X_{t-1} = X_{s-1}$. But since $Tr_t \neq Tr_s$, we have that $AD_j[i] \neq AD_k[i]$ or $P_j[i] \neq P_k[i]$. Then $U_t \neq U_s$, so $Pr(U_t \| V_t = U_s \| V_s) = 0$.

(3) $s-1, t-1$ are equal, and s, t are at different stages. By Lemma A.2, we have $Y_{t-1} = Y_{s-1}$, $R_{t-1} = R_{s-1}$, and $V_t = Y_{t-1} \oplus R_{t-1} \oplus a_t$, $V_s = Y_{s-1} \oplus R_{s-1} \oplus a_s$, $a_t, a_s \in \{0, 1, 3, 5\}$ depend on the stage of t, s. Since s, t are at different stages, $a_t \neq a_s$, so $V_t \neq V_s$ and $Pr(U_t \| V_t = U_s \| V_s) = 0$.

(4) $s-1, t-1$ are not equal. If s is at the initialization stage, then it is same as case (1) that $Pr(U_t \| V_t = U_s \| V_s) \leq 1/2^n$. If it is not, then one of the three holds: $U_s = X_{s-1}$, $U_s = X_{s-1} \oplus AD_k[i]$ for some i, k, $U_s = X_{s-1} \oplus P_k[i]$ for some i, k. We can write that $U_s = X_{s-1} \oplus In_s$, which $In_s = 0^n$ or $AD_k[i]$ or $P_k[i]$. It is the same that $U_t = X_{t-1} \oplus In_t$, which $In_t = 0^n$ or $AD_j[l]$ or $P_j[l]$. Since $Tr_{t-1} \neq Tr_{s-1}$, X_{s-1} and X_{t-1} are uniformly random and independent, and by Lemma A.1, X_{s-1} is independent with In_s, X_{t-1} is independent with In_t. So we have $X_{t-1} \oplus In_t$ is uniformly random and independent with $X_{s-1} \oplus In_s$, and $Pr(U_s = U_t) = 1/2^n$, so $Pr(U_t \| V_t = U_s \| V_s) \leq 1/2^n$.

So in all three cases that $bad'_{t+1} = false$, $Pr(U_t \| V_t \in S_t) \leq |S_t|/2^n$, that is, the probability for $bad^*_t = false$ and $bad^*_{t+1} = true$ is not greater than $|S_t|/2^n$. Then, we have that the event of bad^* set to true has a probability of not greater than $1/2^n + ... + (h + p + 3q - 1)/2^n \leq (h + p + 3q)^2/2^{n+1}$.