The JAMBU Lightweight Authentication Encryption Mode (v2.1)

15 Sep, 2016

Designers: Hongjun Wu, Tao Huang

Submitters: Hongjun Wu, Tao Huang

Contact: `wuhongjun@gmail.com`

Division of Mathematical Sciences
Nanyang Technological University, Singapore

Table of Contents

1 Introduction

Authenticated encryption mode is one of the commonly used method in the design of authenticated ciphers. The ISO/IEC 19772:2009 [9] standardized several modes for authenticated encryption, including EAX [2], CCM [22], GCM [17] and OCB 2.0 [21]. And a number of other authenticated encryption modes have been proposed in the past two decades, *e.g.,* IAPM [12], CWC [14], HBS [11], BTM [10] and McOE [7].

An important trend in the current development of cryptography is to design lightweight cryptographic primitives since the increasing needs for low-cost embedded systems such as RFID tags, sensor networks and smart cards. Several authenticated encryption schemes have been proposed for the lightweight usage, such as Hummingbird-2 [6], ALE [4], and FIDES [3].

However, those above mentioned lightweight authenticated encryption schemes are dedicated design and can not be used as a mode of operation to convert an encryption scheme into an authenticated cipher. Moreover, it turns out that it is quite difficult to construct a secure lightweight authenticated cipher. Security flaws were discovered for ALE and FIDES shortly after their publications [5, 13, 23]. Hence, it is meaningful to develop secure lightweight authenticated encryption modes so that the previous designs of lightweight block ciphers can be converted to lightweight authenticated ciphers.

In this document, we propose a lightweight authenticated encryption mode JAMBU. Then we use the block ciphers SIMON and AES-128 to construct an authenticated ciphers – SIMON-JAMBU and AES-JAMBU.

2 The JAMBU Mode of Operation

2.1 Preliminary

2.1.1 Operations

The following operations are used in JAMBU:
$\oplus$: bit-wise exclusive OR.
$\|$: concatenation.

2.1.2 Notations and Constants

The following notations are used in JAMBU specifications.
0^a : a bit of '0's.
AD : associated data (this data will not be encrypted or decrypted).
$adlen$: bit length of the associated data with $0 \leq adlen < 2^{64}$.
C : ciphertext.
C_i : a ciphertext block (the last block may be a partial block).
E_K : encryption of one block using the secret key K.
IV : initialization vector used in JAMBU.
K : secret key used in JAMBU.

$msglen$: bit length of the plaintext/ciphertext with $0 \leq msglen < 2^{64}$.
m_i : a data block.
n : half of the block size used in JAMBU.
N : number of the associated data blocks and plaintext blocks after padding. $N = N_A + N_P$
N_A : number of the associated data blocks after padding.
N_P : number of the plaintext blocks after padding.
P : plaintext.
P_i : a plaintext block (the last block may be a partial block).
R : an additional state used for encryption. The size is half of the block size.
S : an internal state which will be used for encryption.
T : authentication tag.
t : bit length of the authentication tag

2.2 Parameters

As an authenticated encryption mode, JAMBU accepts the underlying block ciphers with even bits block size which is denoted by $2n$. The key size is the same as the one used in the block cipher. The tag length is n bits. We limit the maximum length of messages to be 2^n bits under a single key.

2.3 Padding

The following padding scheme is used in JAMBU . 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.

2.4 Initialization

JAMBU uses an n-bit initialization vector(IV). The initialization vector (public message number) is public. Each key/IV pair should be used only once to achieve the maximum security of the scheme.

Let (X, Y) represent the composition of n-bit states X and Y which results in a state of $2n$-bit. The initial state is set as $S_{-1} = (0^n, IV)$. The following operations are used for initialization.

1. $(X_{-1}, Y_{-1}) = E_K(S_{-1})$;
2. $R_0 = X_{-1}$;
3. $S_0 = (X_{-1}, Y_{-1} \oplus 5)$.

The initialization of JAMBU is shown in Fig. 1.

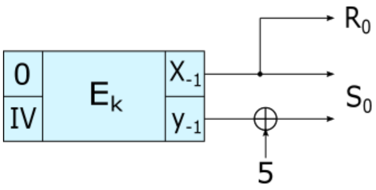

Fig. 1: Initialization of JAMBU .

2.5 Processing the associated data

The associated data is divided into n-bit blocks and processed sequentially. For the last block, the padding scheme is applied to make it a full block. Note that at least one block is processed in the processing of AD. Namely, if the length of AD, *adlen*, is 0, a padded block $1 \parallel 0^{n-1}$ will be processed. Let N_A be the number of AD blocks after padding, the AD is processed as follows.

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

$$
\begin{aligned}
(X_i, Y_i) &= E_K(S_i);\\
U_{i+1} &= X_i \oplus A_i;\\
V_{i+1} &= Y_i \oplus R_i \oplus 1;\\
S_{i+1} &= (U_{i+1}, V_{i+1});\\
R_{i+1} &= R_i \oplus U_{i+1}.
\end{aligned}
$$

Fig. 2 shows the processing of two blocks of associated data.

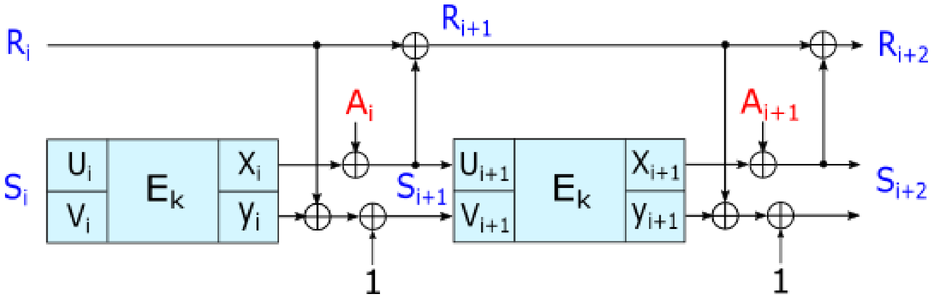

Fig. 2: Processing associated data.

2.6 Encryption of JAMBU

In the encryption of JAMBU, the plaintext is divided into blocks of n-bit. And the last block is padded using the padding scheme specified previously. In each step of the encryption, a plaintext block P_i is encrypted to a ciphertext block C_i.

If the last plaintext block is a full block, a block of "$1||0^{n-1}$" is processed without any output. Fig. 3 shows the encryption of two plaintext blocks.

Let N_P be the number of plaint blocks after padding, the encryption is described as follows:

- For $i = N_A$ to $N_A + N_P - 1$, we perform encryption and update the state:
  $(X_i, Y_i) = E_K(S_i)$;
  $U_{i+1} \quad = X_i \oplus P_{i-N_A}$;
  $V_{i+1} \quad = Y_i \oplus R_i$;
  $S_{i+1} \quad = (U_{i+1}, V_{i+1})$;
  $R_{i+1} \quad = R_i \oplus U_{i+1}$.
  $C_{i-N_A} \quad = P_{i-N_A} \oplus V_{i+1}$ if $i < N_A + N_P - 1$ or the last plaintext block is a partial block; otherwise, C_{N_P-1} will not be computed.
- The final ciphertext block is truncated to the actual length of last plaintext block from the most significant bit side.

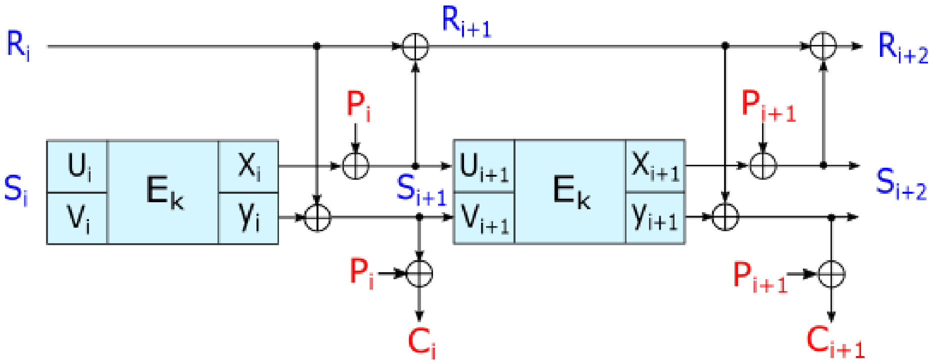

Fig. 3: Processing the plaintext.

2.7 Finalization and tag generation

After all the padded plaintext blocks are processed, suppose the state is S_{N+1} and R_{N+1} ($N = N_A + N_P - 1$), we use following steps to generate the authentication tag, see Fig. 4.

1. $(X_{N+1}, Y_{N+1}) = E_K(S_{N+1})$;
2. $U_{N+2} = X_{N+1}$;
3. $V_{N+2} = Y_{N+1} \oplus R_{N+1} \oplus 3$;
4. $R_{N+2} = R_{N+1} \oplus X_{N+1}$;
5. $S_{N+2} = (X_{N+2}, Y_{N+2})$;
6. Authentication tag is generated as $T = R_{N+2} \oplus X_{N+2} \oplus Y_{N+2}$.

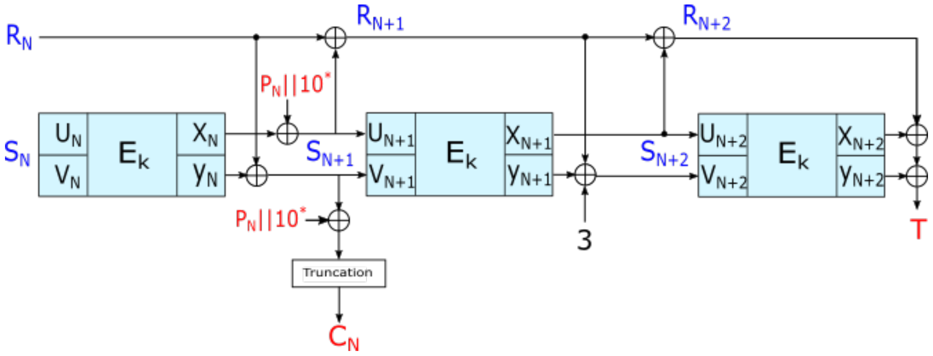

Fig. 4: Finalization and tag generation.

2.8 The decryption and verification

The decryption and verification are similar to the encryption and authentication, except that the ciphertext block is XORed with the sub-state V to compute the plaintext block. For the final block, the ciphertext is padded using the same scheme as the plaintext before the XOR operation. If the length of ciphertext block is a multiple of n, another block of "$1||0^{n-1}$" is processed similar as in the encryption.

A tag T' is generated after the decryption and is compared to the tag T. If the two tags match, the plaintext is outputted.

3 The SIMON-JAMBU and AES-JAMBU authenticated ciphers

To construct a lightweight authenticated cipher using JAMBU, there are many choices of underlying block ciphers. In this specification, we use SIMON [1] as our primary choice of underlying block ciphers. We remark that JAMBU is capable to be used with any other block ciphers, especially for those which are designed for the lightweight applications.

The SIMON-JAMBU authenticated ciphers
In this specification, we take SIMON as our primary choice of the application of JAMBU. SIMON is a family of lightweight blocks cipher published by NSA in 2013. It specifies 10 block ciphers with block sizes 32, 48, 64, 96, 128 bits and key sizes 64, 72, 96, 128, 144, 192, 256 bits in the SIMON family. SIMON uses the Feistel network with simple round operations which are efficient in both hardware and software. The Feistel network can be easily adopted in JAMBU as the internal state of the block cipher is naturally divided into two sub-states. In this document, we will use SIMON64/96, SIMON96/96, SIMON128/128 in our designs of SIMON-JAMBU authenticated ciphers.

As required by the CAESAR competition, we include a description of SIMON block cipher in the Appendix A from the original paper [1] so as to make the SIMON-JAMBU self-contained.

The AES-JAMBU authenticated cipher
We also apply the JAMBU mode to the most widely used block cipher AES-128 and construct an authenticated cipher AES-JAMBU. Although AES itself is not designed for lightweight usage, it is often needed to be employed in constrained devices. AES-JAMBU will be a good choice when authenticated encryption is needed based on AES in such cases for the very small additional area.

Recommended parameter sets

- **Primary recommendation:** SIMON-JAMBU96/96
  96-bit key, 48-bit nonce, 144-bit state, 48-bit tag
  Reason: lightweight state size with reasonable security.
  Use Case 1: Lightweight applications.
  Use Case 3: Defense in depth.
- **Secondary recommendation:** SIMON-JAMBU64/96
  96-bit key, 32-bit nonce, 96-bit state, 32-bit tag
  Reason: very small state size to fit extremely constrained condition.
  Use Case 1: Lightweight applications.
  Use Case 3: Defense in depth.
- **Tertiary recommendation:** SIMON-JAMBU128/128
  128-bit key, 64-bit nonce, 192-bit state, 64-bit tag
  Reason: provide high security level with SIMON128/128.
  Use Case 1: Lightweight applications.
  Use Case 3: Defense in depth.
- **Quaternary recommendation:** AES-JAMBU
  128-bit key, 64-bit nonce, 192-bit state, 64-bit tag
  Reason: provide high security level with widely used AES-128.
  Use Case 3: Defense in depth.
  Use Case 1: Lightweight applications.

4 Security Goals

4.1 Security of JAMBU in a nonce-respecting scenario

The security goals of JAMBU are given in Table 1. The total length of messages (plaintext and associated data) protected by a single key is limited to 2^n bits (for a block cipher with $2n$-bit block size). To achieve the maximum security, each key and IV pair should be used to protect only one message. If verification fails, the new tag and the decrypted ciphertext should not be given as output.

Note that the integrity security in Table 1 includes the integrity security of plaintext, associated data and nonce and under the assumption that κ-bit encryption security is employed and n-bit tag is generated.

Table 1: Security Goals of JAMBU. (κ-bit key, $2n$-bit block size.)

	Confidentiality (bits)	Integrity (bits)
JAMBU	κ	n

4.2 Security of JAMBU in a nonce-misuse scenario

In case that nonce is reused under the same key, the integrity security of JAMBU remains as n-bit. Currently, we proved that when the total message size is less than $2^{n/2}$ bits, the integrity security of JAMBU remains as n-bit.

The confidentiality of JAMBU is partially compromised when nonce is reused. If the first i plaintext blocks are the same, then the $(i+1)$-th and $(i+2)$-th plaintext blocks are insecure. (It is obvious that the security of the $(i+1)$-th plaintext block is insecure when nonce is reused. It was shown in [19] that if the first i plaintext blocks are the same, then the $(i+2)$-th plaintext block is also insecure if the attacker can repeat the attack using the same nonce and chosen plaintexts for $2^{n/2}$ times.)

If the ciphertext is released when verification fails, the security of JAMBU is similar to that of nonce reuse.

5 The Security Analysis of JAMBU

In this section, we analyze the encryption and authentication security of JAMBU.

5.1 Encryption Security of JAMBU

5.1.1 Nonce-Respecting Scenario

The encryption of JAMBU can be seen as a variant of the Cipher Feedback (CFB) mode from the NIST recommendation [18]. The main difference is that in JAMBU the message block and an additional state block R are XORed with the internal state. The encryption of JAMBU is expected to be as strong as the underlying block cipher as long as each key/IV is used to protect only one message. Thus, the plaintext block and the additional state will not affect the randomness of the output of the CFB mode, and the confidentiality of JABMU can be implied from CFB.

5.1.2 Nonce Misuse Scenario

For JAMBU, the encryption has intermediate level of robustness in the nonce reuse circumstances. More specifically, after the identical blocks in the prefix, the first and second message blocks are insecure.

Regarding to the encryption security under IV misuse cases, it is not that meaningful to consider the distinguishing attack, as it can be trivially done. For key recovery attack, it is as difficult as breaking the underlying block cipher. Here we will discuss the plaintext recovery attack without the knowledge of the key.

Suppose that a nonce-misuse chosen plaintext adversary wants to decrypt a secure ciphertext block, say C_{i+2}. If he can find the correct plaintext with probability greater than $1/2^n$, he has a better chance than the random guess. Otherwise the ciphertext is secure. In our setting, the adversary may query messages with common blocks up to P_{i-1} so that the C_{i+2} is secure. To decrypt C_{i+2}, $Y_{i+2} \oplus R_{i+2}$ must be known. Since, $X_{i+2}||Y_{i+2} = E_K(U_{i+2}||V_{i+2})$, if $U_{i+2}||V_{i+2}$ has never been queried before, Y_{i+2} will be random and the adversary can not win the game. Thus, the adversary must be able to obtain a collision of $U_{i+2}||V_{i+2}$. Note that $V_{i+2} = Y_{i+1} \oplus R_{i+1}$ does not have common prefix the any other queries and the value of R_{i+1} is secret, this condition can only be satisfied with probability $1/2^n$. But since C_{i+1} is known and the plaintext can be chosen, it is possible to obtain a collision on V_{i+2}. Suppose that there is some V_j satisfies that $V_{i+2} = V_j$, the probability that $U_{i+2} = U_j$ is $1/2^n$. To see this, we write the condition as $X_{i+1} \oplus P_{i+1} = X_j \oplus P_j$. Since P_{i+1} has unique prefix by our assumption, the value is fixed, and X_{i+1}, X_j are the output of encryption which can not be controlled. The probability that $P_j = P_{i+1} \oplus X_{i+1} \oplus X_j$ is $1/2^n$. Therefore, the probability to obtain a collision of $U_{i+2}||V_{i+2}$ is at most $1/2^n$. Hence, except the first two blocks after the common prefix, the blocks are secure.

5.2 Authentication Security of JAMBU

In this section, we analyze the authentication security of JAMBU. We consider the nonce misuse scenario in our analysis.

Let $\mathcal{A}$ be a nonce misuse adversary. Suppose that the total number of queries $\mathcal{A}$ made is q. We assume that the adversary does not make repeated queries or trivial queries such as decrypting of ciphertext and tag obtained from encryption. For query j, the message block length is l_j. Let the maximum message block length for all queries be $l_{\max}$. Since the padding of JAMBU is injective, we only consider the fully block messages without padding block. For simplicity, we let all the message blocks be plaintext blocks. When the associated data blocks are considered, the analysis will be similar. Let (S_i^j, R_i^j) be the state of before the block cipher encryption calls to encrypt the i-th plaintext block of the j-th query (See Fig. 3). M_i^j indicates the i-th message block in query j. Using the same notation as Section 2, $S_i^j = (U_i^j, V_i^j)$ and $E_K(S_i^j) = (X_i^j, Y_i^j)$. Let **CollR** be the event that there exists $R^j_{l_{j+2}} = R^{j'}_{l'}$ for $1 \leq j, j' \leq q$, $1 \leq l' \leq l_{j'+2}$ and $(j, l_{j+2}) \neq (j', l')$. Let **CollS** be the event that there exists $S^j_{l_j+2} = S^{j'}_{l'}$ for $1 \leq j, j' \leq q$, $1 \leq l' \leq l_{j'+2}$, and $(j, l_{j+2}) \neq (j', l')$.

Then we define event **bad** $=$ **CollR** $\wedge$ **CollS**.

We have the following lemma:

Lemma 1. *When event* **bad** *does not occur, the adversary* $\mathcal{A}$ *has no advantage over a random guess.*

Proof. We consider the tag generation in each query. For query j, $T_j = R^j_{l_j+2} \oplus X_{l_j+2} \oplus Y_{l_j+2}$. Given that the event **bad** does not occur, either $R^j_{l_j+2}$ is unknown to the adversary or the output state of the encryption (X_{l_j+2}, Y_{l_j+2}) is unknown to the adversary. In both case, the best the adversary can do is to make a random guess on the tag value. ☐

Now we need to bound the probability of the event **bad**.

Lemma 2. *Suppose that an adversary makes* q *queries and each query has maximum message length* l_{max}*, the probability of event* **bad** *occurs is:*

$$\boldsymbol{Pr}(\textsf{\textbf{bad}}) \leq \frac{3q^2 l_{max}^2}{2^{2n}} + \frac{2q^2 l_{max}}{2^{2n}} + \frac{5q^2}{2^{2n+1}} - \frac{q(l_{max}+2)}{2^{2n+1}}$$

Proof. First, we consider an event **CollIn** such that there exists internal state collisions $(S^j_i, R^j_i) = (S^{j'}_{i'}, R^{j'}_{i'})$ for $1 \leq i < l_j + 2$, $1 \leq i' < l_{j'} + 2$, $1 \leq j, j' \leq q$, $(i, j) \neq (i', j')$ and $M^j_{1,\ldots,i} \neq M^{j'}_{1,\ldots,i'}$.

A general approach to construct an internal collision is from the birthday attack. This was discussed by Preneel and van Oorschot [20], in which it was shown that all iterated MACs with n-bit internal state can be attacked with $O(2^{n/2})$ queries. For JAMBU, the internal state size is $3n$ bits. Thus, around $2^{3n/2}$ messages are needed for an internal collision using the birthday attack.

Assume that event **CollIn** occurs, we will decide the probability. Suppose we have the first internal collision for query j and j' such that $(R^j_{i+1}, U^j_{i+1}, V^j_{i+1}) = (R^{j'}_{i'+1}, U^{j'}_{i'+1}, V^{j'}_{i'+1})$ (Here we write S as (U, V)). We have $R^j_{i+1} = R^j_i \oplus U^j_{i+1}$, $U^j_{i+1} = X^j_i \oplus M^j_i$, and $V^j_{i+1} = Y^j_i \oplus R^j_i$, and the similar expressions holds for the states at query j'. Hence, we can derive the following necessary conditions for the internal collision:

$$R^j_i = R^{j'}_{i'} \tag{1}$$

$$Y^j_i = Y^{j'}_{i'} \tag{2}$$

$$X^j_i \oplus X^{j'}_{i'} = M^j_i \oplus M^{j'}_{i'} = \delta \neq 0 \tag{3}$$

Note that condition (3) is non-zero since otherwise another internal state collision would occur in the previous blocks. We consider two cases:

(a) The adversary does not known δ, so the condition (3) is satisfied with probability $1/2^n$. The probability of condition (2) is $1/2^n$ as they are just the output of an ideal block cipher with a difference in the input. In this case, the probability of collision for the pair of internal state is bounded by $1/2^{2n}$.

(b) The adversary obtained the difference δ for $M^j_{1,\ldots,i-1}$ and $M^{j'}_{1,\ldots,i'-1}$. This is possible in the nonce reuse cases using the method proposed in [19]. Now we consider condition (1). If $U^j_i = U^{j'}_{i'}$, condition (1) implies $X^j_{i-1} \oplus X^{j'}_{i'-1} = M^j_{i-1} \oplus Mj'_{i'-1}$ and $Y^j_{i-1} \neq Y^{j'}_{i'-1}$. Note that $X^j_{i-1} \oplus X^{j'}_{i'-1} = M^j_{i-1} \oplus Mj'_{i'-1}$ holds with probability $1/2^n$. On the other hand, If $U^j_i \neq U^{j'}_{i'}$, $M^j_{i-1} \oplus Mj'_{i'-1} =$

$X^j_{i-1} \oplus X^{j'}_{i'-1} \oplus R^j_{i-1} \oplus R^{j'}_{i'-1}$ which has probability $1/2^n$. Together with condition (2), the probability of collision for the pair of internal state is bounded by $1/2^{2n}$ in this case.

From the above analysis,

$$\mathbf{Pr}(\textsf{\textbf{CollIn}}) \leq \frac{(l_{\max}+1)^2 q^2}{2^{2n}} - \frac{(l_{\max}+1)q}{2^{2n}} \tag{4}$$

Now consider the conditional probability of event **bad** on event **CollIn**. Then we have

$$\mathbf{Pr}(\textsf{\textbf{bad}}) \leq \mathbf{Pr}(\textsf{\textbf{bad}} \mid \neg\textsf{\textbf{CollIn}}) + \mathbf{Pr}(\textsf{\textbf{CollIn}}) \tag{5}$$

To bound $\mathbf{Pr}(\textsf{\textbf{bad}} \mid \neg\textsf{\textbf{CollIn}})$, we consider the tag generation in each query. For query j, $T_j = R^j_{l_j+2} \oplus X_{l_j+2} \oplus Y_{l_j+2}$. Given that the event **bad** does not occur, either $R^j_{l_j+2}$ is unknown to the adversary or the output state of the encryption (X_{l_j+2}, Y_{l_j+2}) is unknown to the adversary. In both case, the best the adversary can do is to make a random guess on the tag value.

Consider the event **CollS** first. Given that $S^j_{l_j+2} = S^{j'}_{l'}$, it can be divided into two cases: $l' \neq l_{j'}+2$ and $l' = l_{j'}+2$.

Case 1: $l' \neq l_{j'}+2$. It indicates that $S^j_{l_j+2}$ collides with a state S which is not the final state of a query. In this case, the following conditions holds:

$$X^j_{l_j+1} = X^{j'}_{l'-1} \tag{6}$$

$$Y^j_{l_j+1} \oplus Y^{j'}_{l'-1} = R^j_{l_j+1} \oplus R^{j'}_{l'-1} \oplus 3 \tag{7}$$

We further consider two sub-cases.

> *Subcase A*: $S^j_{l_j+1} = S^{j'}_{l'-1}$. Under this condition, $R^j_{l_j+1} \oplus R^{j'}_{l'-1} = 3$ holds. Then $M^j_{l_j} \oplus M^{j'}_{l'-2} = R^j_{l_j} \oplus R^{j'}_{l'-2} \oplus 3$. Given that $\neg$**CollIn** occurs and no repeated queries are made, $\mathbf{Pr}((Y^j_{l_j} \oplus Y^{j'}_{l'-2} = 3) \wedge (X^j_{l_j} \oplus X^{j'}_{l'-2} = M^j_{l_j} \oplus M^{j'}_{l'-2})) = 1/2^{2n}$.
>
> *Subcase B*: $S^j_{l_j+1} \neq S^{j'}_{l'-1}$. Under this condition, the input of a block cipher encryption has difference and the output difference is fixed by equation (1) and (2). Hence, the probability is $1/2^{2n}$.

Summing up the above probabilities, the probability for this case is upper bounded by $q^2(l_{\max}+1)/2^{2n}$.

Case 2: $l' = l_{j'}+2$ indicates that $S^j_{l_j+2}$ collides with a state S which is the final state of a query. Consider the state R^j and $R^{j'}$.

When $R^j_{l_j+2} = R^{j'}_{l_{j'}+2}$, immediately we obtain an internal state collision $(S^j_{l_j+1}, R^j_{l_j+1}) = (S^{j'}_{l_{j'}+1}, R^{j'}_{l_{j'}+1})$, which violates the assumption that $\neg$**CollIn**.

When $R^j_{l_j+2} \neq R^{j'}_{l_{j'}+2}$, let $R^j_{l_j+2} \oplus R^{j'}_{l_{j'}+2} = \delta$. Then, we have $(Y^j_{l_j+1} \oplus Y^{j'}_{l_{j'}+1} = \delta)$ and $((X^j_{l_j+1} \oplus X^{j'}_{l_{j'}+1} = 0$. The probability is $1/2^{2n}$ for these two conditions to hold.

From above analysis,

$$\mathbf{Pr}(\textsf{bad} \mid \neg\textsf{CollIn}) \leq \frac{2q^2 l^2_{\max}}{2^{2n}} + \frac{3q^2}{2^{2n+1}} - \frac{q}{2^{2n+1}} \tag{8}$$

From equation (4), (5) and (8) we get

$$\mathbf{Pr}(\textsf{bad}) \leq \frac{3q^2 l^2_{\max}}{2^{2n}} + \frac{2q^2 l_{\max}}{2^{2n}} + \frac{5q^2}{2^{2n+1}} - \frac{q(l_{\max}+2)}{2^{2n+1}} \tag{9}$$

□

From the Lemma 1 and Lemma 2, we have the following theorem for the authentication security of JAMBU.

Theorem 1. *For a nonce misuse adversary $\mathcal{A}$ making at most q queries to JAMBU with at most l_{max} blocks of message in each query, the advantage that the adversary can make a successful forge over random guess has the following bound*

$$\boldsymbol{Adv}^{auth}_{JAMBU} \leq \frac{3q^2 l^2_{max}}{2^{2n}} + \frac{2q^2 l_{max}}{2^{2n}} + \frac{5q^2}{2^{2n+1}} - \frac{q(l_{max}+2)}{2^{2n+1}}.$$

Theorem 1 shows that the authentication security of JAMBU is n-bit when the total message size is less than $2^{n/2}$ bits.

6 Features

- Lightweight. In addition to the registers used in the underlying block cipher, the JAMBU authenticated encryption mode only requires one additional register with half of the block size. For AES-GCM, two additional registers[1] are needed and each has equal length as the block size. And for fast implementation of GCM operations, a look up table is very helpful. However, when the table is used, a much larger amount of memory will be needed. It makes AES-GCM not suitable for lightweight implementations.
- Partial resistance against IV reuse. When the IV is accidentally reused under the same key, the security of encryption and authentication is not completely compromised. Notice that in AES-GCM, the nonce reuse will lead to the lost of all confidentially and integrity.

---

[1] The two registers are used to: store the length of P and AD; store the chaining value for authentication.

7 Performance

7.1 Hardware performance

The design of JAMBU is hardware-oriented. In the hardware implementation of authenticated ciphers, the state size is an important factor, especially for low-cost embedded systems. To compare the hardware efficiency of the authenticated encryption modes in term of area, We look at the state size when an authenticated encryption mode is applied to a $2n$-bit block cipher. We compare the state size in JAMBU with the existing authentication modes. The results are given in Table 3. As a lightweight authenticated encryption mode, JAMBU provides the minimum state size for the hardware implementation.

Table 2: The comparison (in state size) for authenticated encryption modes, assuming the underlying block cipher has block size $2n$ bits

Modes	State size	Increments
CCM	$4n$	$2n$
GCM	$6n$	$4n$
OCB3	$6n$	$4n$
EAX	$8n$	$6n$
COPA	$6n$	$4n$
CPFB	$6n$	$4n$
ELmD	$8n$	$6n$
SILC	$4n$	$2n$
CLOC	$4n$	$2n$
JAMBU	$3n$	n

We implemented SIMON-JAMBU96/96 using the CAESAR hardware API proposed by Homsirikamol*et al.* from GMU [8]. On modern FPGA Vertix-7, the frequency of SIMON-JAMBU96/96 is 434 MHz, using 375 slices (1254 LUTs) in area. The throughput of SIMON-JAMBU96/96 for long message is 385 Mbits/s.

7.2 Software performance

We implemented SIMON-JAMBU and AES-JAMBU in C code, the AES instruction is used in AES-JAMBU. We tested the speed on the Intel Core i7-4770 3.4GHz processor (Haswell) running 64-bit Linux 14.04. The turbo boost is turned off, so the CPU runs at 3.4GHz in the experiment. The compiler being used is gcc 4.8.2, and the options “-O3 -msse2 -maes” are used. The test is performed by encrypting/decrypting a message repeatedly, and printing out the final message. To ensure that the tag generation is not removed during the compiler optimization process, we use the tag as the IV for processing the next message. To ensure that the tag verification is not removed during the compiler optimization process, we sum up the number of failed verifications and print out the final result.

We tested the speed of CTR, OCB3, GCM, CCM (AES-128 is used in these modes) on the same machine for comparison. The testing programs of CTR, OCB3, GCM and CCM are downloaded following the description given Krovetz and Rogaway in the OCB3 paper [16] and their website [15]. The performance comparison is given in Table 3.

For 4096-byte messages, the speed of SIMON-JAMBU64/96 is 51.94 cpb which is about two times of the SIMON64/96 speed, 27.3 cpb, mentioned in [1]. The speed of AES-JAMBU is about 11.6 cpb.

Table 3: The software speed comparison (in cycles per byte) for different message length on Intel Haswell.

	64B	128B	256B	512B	1024B	4096B
AES-128-CTR	1.71	1.52	1.13	1.09	1.00	0.97
AES-128-CCM	6.62	5.56	5.03	4.76	4.63	4.53
AES-128-GCM	5.93	3.84	2.91	2.46	2.24	2.07
AES-128-OCB3	3.46	2.15	1.43	1.09	0.93	0.78
SIMON-JAMBU64/96	83.24	62.78	57.21	54.79	53.21	51.94
SIMON-JAMBU96/96	124.72	95.67	84.93	79.67	76.93	75.08
SIMON-JAMBU128/128	76.11	58.26	49.55	45.61	43.06	41.45
AES-JAMBU	24.41	17.08	13.41	11.57	10.65	9.98

8 Design rationale

JAMBU is designed to be a lightweight authenticated encryption mode which can offer partial resistance against IV reuse.

To make this mode lightweight, we introduces only an n-bit extra register for a $2n$-bit block size. And we only use the bit-wise XOR operations in the JAMBU mode.

The padding scheme used in JAMBU does not require the length information to be stored in a register. This reduces the memory requirements.

To offer a certain level of security against IV reuse, we use a block cipher encryption in the state update and only half of the state is leaked after encryption. The plaintext is injected into the other half of the state which is unknown to the attacker.

Several constants are XORed with the state in JAMBU. They are used to separate the initialization, associate data processing, plaintext processing, and finalization.

SIMON-JAMBU takes advantage of the lightweight block cipher SIMON. It can achieve a very lightweight hardware implementation.

AES-JAMBU uses AES as the underlying block cipher. It can take advantages from the security analysis AES as well as the fast implementation of AES using AES-NI.

9 Changes

9.1 Changes from v2 to v2.1

1. There is no tweak to the JAMBU mode.
2. We have re-written the security analysis of JAMBU, which gives more details of the security bound of JAMBU authentication.
3. Hardware performance data of SIMON-JAMBU96/96 on FPGA is added.
4. Some editorial changes.

9.2 Changes from v1 to v2

1. There is no tweak to the JAMBU mode.
2. We gave more security analysis of the JAMBU mode in Section 5 and Section 6.
3. The lightweight block cipher SIMON is added to the recommended underlying block ciphers used in JAMBU mode in Section 3. And the software performance of SIMON-JAMBU is included in Section 8. In fact, any secure block cipher can be used in the JAMBU mode.
4. The security claim on encryption security for nonce reuse is slightly changed in Section 4. The new claim is that if nonce is reused, and if the first i blocks are the same, then the security of the $(i+1)$-th and $(i+2)$-th plaintext blocks are insecure. (In the v1 document, we claimed that if nonce is reused, and if the first n blocks are the same, then the security of the $(i+1)$-th block is insecure.)

10 Intellectual property

JAMBU is not patented and it is free of intellectual property restrictions. If any of this information changes, the submitter/submitters will promptly (and within at most one month) announce these changes on the crypto-competitions mailing list.

11 Consent

The submitter/submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitter/submitters understand that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitter/submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitter/submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitter/submitters understand that if they disagree with published analyses then they are expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitter/submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

References

1. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.
2. M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In *Fast Software Encryption*, pages 389–407. Springer, 2004.
3. B. Bilgin, A. Bogdanov, M. Knežević, F. Mendel, and Q. Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In *Cryptographic Hardware and Embedded Systems-CHES 2013*, pages 142–158. Springer, 2013.
4. A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser. ALE: AES-Based Lightweight Authenticated Encryption. In *Fast Software Encryption*, 2013.
5. I. Dinur and J. Jean. Cryptanalysis of FIDES. In *Fast Software Encryption–FSE 2014*. Springer, 2014.
6. D. Engels, M.-J. O. Saarinen, P. Schweitzer, and E. M. Smith. The Hummingbird-2 Lightweight Authenticated Encryption Algorithm. In *RFID. Security and Privacy*, pages 19–31. Springer, 2012.
7. E. Fleischmann, C. Forler, and S. Lucks. McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In *Fast Software Encryption–FSE 2012*, pages 196–215. Springer, 2012.
8. E. Homsirikamol, W. Diehl, A. Ferozpuri, F. Farahmand, P. Yalla, J.-P. Kaps, and K. Gaj. CAESAR Hardware API. Cryptology ePrint Archive, Report 2016/626, 2016. `http://eprint.iacr.org/2016/626`.
9. ISO/IEC 19772:2009. *Information technology – Security techniques – Authenticated encryption.* ISO, Geneva, Switzerland, 2009.
10. T. Iwata and K. Yasuda. BTM: A Single-Key, Inverse-Cipher-Free Mode for Deterministic Authenticated Encryption. In *Selected Areas in Cryptography – SAC 2009*, pages 313–330. Springer, 2009.
11. T. Iwata and K. Yasuda. HBS: A Single-Key Mode of Operation for Deterministic Authenticated Encryption. In *Fast Software Encryption–FSE 2009*, pages 394–415. Springer, 2009.
12. C. S. Jutla. Encryption Modes with Almost Free Message Integrity. In *Advances in Cryptology – EUROCRYPT 2001*, pages 529–544. Springer, 2001.

13. D. Khovratovich and C. Rechberger. The LOCAL attack: Cryptanalysis of the authenticated encryption scheme ALE. In *Selected Areas in Cryptography – SAC 2013*. Springer Berlin Heidelberg, 2013.
14. T. Kohno, J. Viega, and D. Whiting. CWC: A High-Performance Conventional Authenticated Encrption Mode. In *Fast Software Encryption – FSE 2004*, pages 408–426. Springer, 2004.
15. T. Krovetz and P. Rogaway. Authenticated-encryption software performance: Comparison of ccm, gcm, and ocb. Available at `http://www.cs.ucdavis.edu/~rogaway/ocb/performance/`.
16. T. Krovetz and P. Rogaway. The Software Performance of Authenticated-Encryption Modes. In *Fast Software Encryption – FSE 2011*, pages 306–327. Springer, 2011.
17. D. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM). `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf`.
18. NIST. Recommendation for Block Cipher Modes of Operation-Methods and Techniques. NIST special publication 800–38A, 2001 Edition.
19. T. Peyrin, S. M. Sim, L. Wang, and G. Zhang. Cryptanalysis of JAMBU. In *Fast Software Encryption – FSE 2015*, pages 264–281. Springer, 2015.
20. B. Preneel and P. van Oorschot. MDx-MAC and building fast MACs from hash functions. In *Advances in Cryptology – CRYPTO' 95*, volume 963 of *LNCS*, pages 1–14. Springer, 1995.
21. P. Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In *Advances in Cryptology–ASIACRYPT 2004*, pages 16–31. Springer, 2004.
22. D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). Available from `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmo-des/ccm/ccm.pdf`, 2003.
23. S. Wu, H. Wu, T. Huang, M. Wang, and W. Wu. Leaked-State-Forgery Attack against the Authenticated Encryption Algorithm ALE. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 377–404. Springer Berlin Heidelberg, 2013.

A The specification of SIMON family of block ciphers

SIMON is a family of lightweight blocks cipher published by NSA in 2013. It specifies 10 block ciphers with block sizes 32, 48, 64, 96, 128 bits and key sizes 64, 72, 96, 128, 144, 192, 256 bits in the SIMON family. SIMON uses the Feistel network with simple round operations which are efficient in both hardware and software. Hence the state is denoted as Simon$2n$ for n-bit word. And SIMON$2n/mn$ refers to $2n$-bit internal state with mn-bit key. Note that the key size is always a multiple of the word size n and the value of m can be 2, 3, or 4. In this specification, we only describe SIMON64/96, SIMON96/96 and SIMON128/128 which are used to instantiate JAMBU.

Round function:

SIMON$2n$ encryption makes use of the following n-bit operations:

1. bitwise XOR, $\oplus$,

2. bitwise AND, &, and
3. left circular shift, S^j by j bits.

For round key $k \in GF(2)^n$, the round function is the two-stage Feistel map $R_k : GF(2)^n \times GF(2)^n \to GF(2)^n \times GF(2)^n$ define by:

$$R_k(x, y) = (y \oplus f(x) \oplus k, x),$$

where $f(x) = (Sx \& S^8 x) \oplus S^2 x$.

Number of rounds:

For SIMON64/96, the number of rounds is 42.
For SIMON96/96, the number of rounds is 52.
For SIMON128/128, the number of rounds is 68.

Key schedule:

The SIMON key schedules take a key and from it generate a sequence of T key words $k_0, ..., k_{T-1}$, where T is the number of rounds. The round key generation function depends on the value m for key size mn. The round key (only the cases used in this specification are provided here) can be computed as follows:

- For $k_0, ..., k_{m-1}$, the original m key words are used.
- For k_{i+m} $(0 \leq i < T - m)$,

$$k_{i+m} = c \oplus (z_2)_i \oplus k_i \oplus (I \oplus S^{-1}) S^{-3} k_{i+1}, \text{ if } m = 2,$$

$$k_{i+m} = c \oplus (z_2)_i \oplus k_i \oplus (I \oplus S^{-1}) S^{-3} k_{i+2}, \text{ if } m = 3,$$

where $c = 2^n - 4$ is a constant, I is the identity and $z_2(j)$ is the j^{th} bit of constant

$$z_2 = 10101111011100000011010010011000101000010001111110010110110011.$$