JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU

Hongjun Wu, Tao Huang

Division of Mathematical Sciences
School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore
{wuhj,huangtao}@ntu.edu.sg

Abstract. In this paper, we propose a lightweight authenticated encryption mode JAMBU. It only needs n-bit extra register for a block cipher with $2n$-bit block size. It achieves n-bit authentication security when 2^n bits are processed under a single key. When nonce (IV) is reused, the encryption security is similar to that of the CFB mode while the message authentication maintains strong security. We instantiate JAMBU with AES-128 as the underlying block cipher to construct an authenticated cipher AES-JAMBU. There are more lightweight authenticated ciphers can be constructed by applying JAMBU with lightweight block ciphers such as SIMON. To the best of our knowledge, JAMBU is the most lightweight authentication mode which provides n-bit authentication security.

Key words: Lightweight authenticated mode, authenticated encryption, JAMBU

1 Introduction

Authenticated encryption mode is one of the commonly used method in the design of authenticated ciphers. The ISO/IEC 19772:2009 [18] standardized several modes for authenticated encryption, including EAX [3], CCM [36], GCM [28] and OCB 2.0 [34]. And a number of other authenticated encryption modes have been proposed in the past two decades, *e.g.,* IAPM [23], CWC [25], HBS [22], BTM [21] and McOE [13]. The ongoing CAESAR competition [8] attracted more designs on the authentication encryption modes, including CBA [17], COPA [1], CPFB [30], OTR [29], CLOC [19], ELmD [9] iFeed [38], JAMBU [16], SCREAM [14] SHELL [35] and SILC [20].

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 [12], ALE [6], and FIDES [4]. 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 [11, 24, 37].

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 encryption schemes. In this paper, we present a lightweight authenticated encryption mode JAMBU and use AES-128 as the underlying block cipher to construct an authenticated cipher – AES-JAMBU.

Among the authenticated encryption modes proposed in the CAESAR competition, SILC and JAMBU are the only two lightweight authenticated modes for block ciphers. Both of the two modes need to call the encryption twice to process one plaintext block. Therefore, the software performances of these two modes are expected to be comparable. The security claims are also similar with nonce-misuse resistance for the authentication. But SILC requires $4n$ bits state for a block cipher with $2n$ bits state which is larger than the $3n$ bits state size of JAMBU.

This paper is organized as follows. In Section 2, we give the specification of JAMBU mode. In Section 3, we present the AES-JAMBU authenticated cipher. In Section 4, we discuss the applications of JAMBU with lightweight block ciphers. The security goals are given in Section 5, followed by the security analysis of AES-JAMBU under nonce-respecting and nonce-misuse scenarios in Section 6 and Section 7. The features of JAMBU are presented in Section 8. The software and hardware performance are provided in Section 9. We illustrate the motivation and rationale of our design in Section 10. Finally, Section 11 concludes this paper.

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 accept the underlying block ciphers with even bits block size which is put as $2n$. 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. And 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.

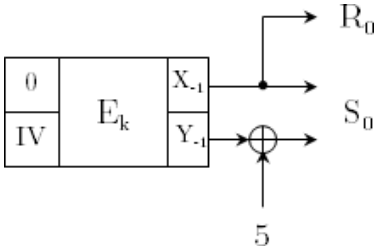

Fig. 1: Initialization of JAMBU .

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.

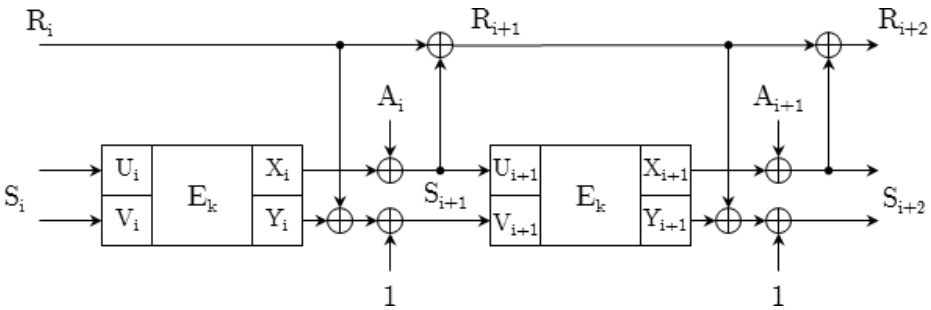

Fig. 2: Processing associated data.

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:
  $(X_i, Y_i) = E_K(S_i)$;
  $U_{i+1} = X_i \oplus P_{i-N_A}$;
  $V_{i+1} = Y_i \oplus R_i$;
  $S_{i+1} = (U_{i+1}, V_{i+1})$;
  $R_{i+1} = R_i \oplus U_{i+1}$.
  $C_{i-N_A} = P_{i-N_A} \oplus V_{i+1}$ 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.

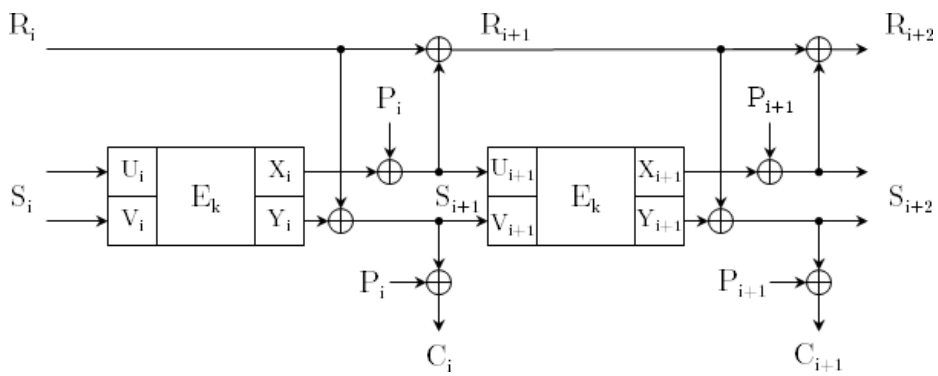

Fig. 3: Processing the plaintext.

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 $T = R_{N+2} \oplus X_{N+2} \oplus Y_{N+2}$.

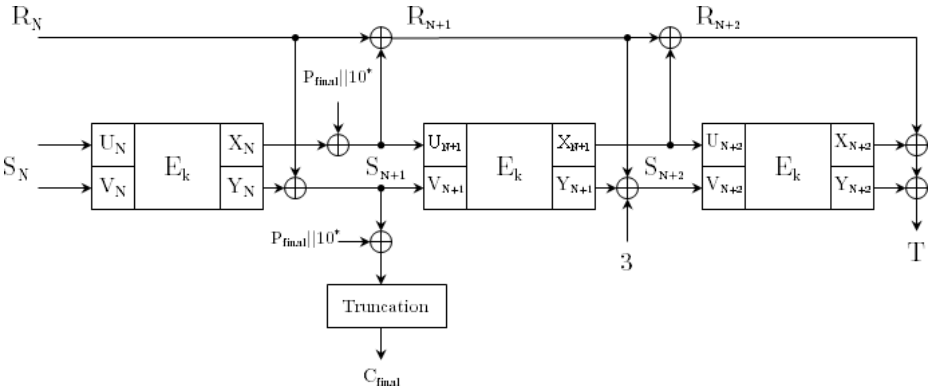

Fig. 4: Finalization and tag generation.

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 AES-JAMBU authenticated cipher

In this section, we will apply the JAMBU mode to the most widely used block cipher AES-128 and construct an authenticated cipher AES-JAMBU.

Recommended parameter set In AES-JAMBU, we recommend specific values for the parameters used in JAMBU. The block size is 128-bit and the size of n is 64-bit. The tag length t is 64-bit and the maximum number of message bits protected by one key is 2^{64} bits.

The parameters of AES-JAMBU are summarized in Table 1.

4 The Applications of JAMBU with Lightweight Block Ciphers

To construct a lightweight authenticated cipher using JAMBU, there are many choices of underlying block ciphers which are designed for the lightweight application, *e.g.,* PRESENT [5], KATAN [10], PRINCE [7], LED [15] and SIMON [2].

In this paper, we take SIMON as an instance of the application of JAMBU. SIMON is a family of lightweight blocks cipher published by NSA in 2013. It

Table 1: Parameter set of AES-JAMBU.

Parameters	Length in bits
Plaintext (P)+Associated data (AD)	$< 2^{64}$
Key (K)	128
Tag (T)	64
Initialization vector (IV)	64
State (S)	320[a]

[a] 128-bit block and 128-bit key are used in AES-128. An additional 64-bit state is used in AES-JAMBU.

specifies 10 block ciphers with block sizes 32, 48, 64, 96, 128 bits and key sizes 64, 72, 96, 128, 144, 192, 256 bits. 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.

For the extremely lightweight case, SIMON32/64 can be used in JAMBU and the internal state size can be as small as 112 bits. For more commonly applications, SIMON64/96 and SIMON96/96 can be good choices and there internal state sizes are 192 bits and 240 bits respectively.

5 Security Goals

The security goals of JAMBU are given in Table 2. There is no secret message number. The public message number is a nonce. 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.

However, in case that the nonce is reused under the same key, the confidentiality of JAMBU is partially compromised. According to the analysis in [32] by Peyrin *et al.* , if the first i plaintext blocks are the same, then the $(i+1)$-th and $(i+2)$-th plaintext blocks are insecure. And the integrity of JAMBU will remain n-bit.

Notice that the integrity security in Table 2 includes the integrity security of plaintext, associated data and nonce and under the assumption that a block cipher with $2n$-bit block size and κ-bit encryption security is employed and n-bit tag is generated. The table also assumes that the total length of message (plaintext and associated data) protected by a single key is limited to 2^n bits.

Table 2: Security Goals of JAMBU.

	Confidentiality (bits)	Integrity (bits)
JAMBU	κ	n

6 The Security Analysis of JAMBU under Nonce-Respecting Scenario

In this section, we analyze the security of JAMBU which includes the security of encryption and the security of authentication when the nonce is unique under the same key.

6.1 The security of encryption

The encryption of JAMBU can be seen as a variant of the Cipher Feedback (CFB) mode from the NIST recommendation [31]. 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.

6.2 The security of message authentication

In this section, we will give our security analysis on message authentication of JAMBU from the key/state recovery attack and the probability for a successful forgery in the nonce-respecting scenario.

6.2.1 Key/state recovery attack

Since the secret key of is protected by the underlying block cipher which is considered ideal, JAMBU is not vulnerable to the key recovery attack. And for state recovery attack, JAMBU also has a strong resistance as there are $2n$-bit unknown state at each step and the secret key is used in the encryption of each step.

6.2.2 The forgery attack

The tag of JAMBU is generated by XORing three n-bit words in the internal state: R_{N+2}, X_{N+2} and Y_{N+2}. Suppose an adversary makes q queries with at most m blocks message. In a forgery attack, the goal of the adversary is to produce a valid tag for a (AD, C) which has never been queried . We consider two cases:

Case 1: the internal state (R, U, V) does not collide with any previous queries before the finalization. Then at least one of the R_{N+1}, U_{N+1} and V_{N+1} is new. Then we discuss the input of the final encryption (U_{N+2}, V_{N+2}):

- (U_{N+2}, V_{N+2}) is new. Then, the final encryption would have a new input, then the tag is valid with probability $1/2^n$.
- (U_{N+2}, V_{N+2}) has been queried before. Then the corresponding R_{N+1} must have difference, otherwise, the input would be a collision which violates our assumption. Therefore, there will be input difference at (U_{N+1}, V_{N+1}) and the output difference must be a fixed value. Then the probability is bounded by $(q-1)/2^{2n} \leq 1/2^n$ when $q \leq 2^n$.

Case 2: the internal state (R, U, V) has collision with some previous queries before the finalization. A general approach to construct an internal collision is from the birthday attack. This was discussed by Preneel and van Oorschot [33] which showed 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 birthday attack. When the number of blocks encrypted is $2^{n-log(n)}$ (the maximum value defined in the specification), there are around $2^{2n-2log(n)}$ pairs of internal states. So the collision probability is around $2^{-3n+2n-2log(n)} = 2^{-n-2log(n)}$ which is less than $1/2^n$.

We can derive the necessary conditions for an internal collision. Suppose we have the first internal collision such that $(R_{i+1}, U_{i+1}, V_{i+1}) = (R_{j+1}, U_{j+1}, V_{j+1})$. We have $R_{i+1} = R_i \oplus U_{i+1}$, $U_{i+1} = X_i \oplus P_i$, and $V_{i+1} = Y_i \oplus R_i$, and the similar expressions holds for the states at step $j+1$. Hence, we can derive the following necessary conditions for the internal collision:

$$Y_i = Y_j \tag{1}$$

$$R_i = R_j \tag{2}$$

$$X_i \oplus X_j = P_i \oplus P_j \tag{3}$$

Note that the condition (1) and (2) imply $V_{i+1} = V_{j+1}$ which can be observed from the keystream block. By generating around $2^{n/2}$ blocks of keystream, we are expected to observe one collision on the keystream block. But this collision is only a necessary condition, and we still need either (1) or (2) holds.

For condition (1), Y_i and Y_j are the output of AES encryption. Without the collision on the input (U_i, V_i) and (U_j, V_j), $Y_i = Y_j$ has probability 2^{-n}. On the other hand, if there is collision on (U_i, V_i) and (U_j, V_j), we get an internal collision on previous state (R_i, U_i, V_i) and (R_j, U_j, V_j) which violates our assumption.

For condition (2), both R_i and R_j are unknown and during the encryption. So the condition (2) can only be satisfied probabilistically, which is $1/2^n$.

For condition (3), when there is no difference on X_i and X_j, it will lead to a collision in previous internal state, which is assumed impossible. And if there is

difference, the difference is uncontrollable by an attacker. Hence, the probability for condition (3) is $1/2^n$.

Given the above analysis, the probability for internal collision given the collision on keystream is $1/2^{2n}$. Therefore, even if we can detect the maximum $2^{(n-log(n))\times 2}/(2^n) = 2^{n-2log(n)}$ collisions on keystream blocks, the probability that an internal collision occurs is less than $1/2^n$.

Hence, the probability of a successful forgery is upper bounded by the the sum of probability in the above two cases, which is $2/2^n$. Hence, JAMBU is strong against the attacks on message authentication.

7 The Security Analysis of JAMBU under Nonce Misuse Scenario

In this section, we analyze the security of AES-JAMBU which includes the security of encryption and the security of authentication when the nonce is unique under the same key.

7.1 The security of encryption

For JAMBU, the encryption still have 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 distinguish 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, we have $P_j = P_{i+1} \oplus X_{i+1} \oplus X_j$ has probability $1/2^n$. 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.

7.2 The security of message authentication

Like the nonce misuse scenario, we consider two cases: *Case 1:* the internal state (R, U, V) does not collide with any previous queries before the finalization. Same analysis applies here and the probability to obtain a valid tag is $1/2^n$.
Case 2: the internal state (R, U, V) has collision with some previous queries before the finalization. The three necessary conditions for internal collision still hold, but the analysis would have some difference in the nonce misuse setting.

First, it is obvious that the internal collisions can happen with identical prefix and a same input block, but this trivial internal collision will not lead to a valid forgery. So we are only interested in the non-trivial internal collisions which do not have common prefix.

Next, if the first difference appears in the i-th block, then it is impossible to eliminate it in the $(i+1)$-th block. This is easy to see as either R_{i+1} or U_{i+1} will have difference.

For condition (1), the probability for a single collision is the same as the unique nonce case. But the difference is that when a collision of Y is found, it can be fixed by using the same prefix when the other conditions are considered. Thus, if an adversary queried at most m blocks of messages, he is expected to find $m(m-1)/2 \times 2^{-n}$ collisions on Y.

For condition (2), since the collision blocks will not have the identical prefix, this conditions has probability 2^{-n} as we discussed in previous analysis.

For condition (3), when the nonce is reused, it is possible to choose the difference of the message block. Then if the difference on X_i and X_j is known, condition (3) can be satisfied with probability 1. According to the method used in [32], the difference in X and R can be derived by $6 \cdot 2^{n/2}$ queries under nonce reused assumption. Here we use $2^{n/2}$ as the lower bound of the number of queries need to find the difference in X and R.

Now if we find N_c collisions of Y which satisfies condition (1), the probability of an internal collision can be computed as $N_c/2^n$ with $N_c \times 2^{n/2}$ queries. On the other hand when N_c random queries are made, the probability of a successful forgery has probability $N_c/2^n$. Therefore, when $N_c \times 2^{n/2}$ queries are made, the probability of internal collision is less than the probability of the trivial attack.

Hence, the probability for a successful forgery is bounded by $2/2^n$.

8 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.

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

- 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.

9 Performance

9.1 Hardware performance

The core feature 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 on the 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, and the results are given in Table 4. As a lightweight authenticated encryption mode, JAMBU provides the minimum state size for the hardware implementation.

Table 3: 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	$4n$	$2n$
COLC	$4n$	$2n$
JAMBU	$3n$	n

9.2 Software performance

We implemented AES-JAMBU in C code using the AES instruction. We tested the speed on the Intel Core i5-2540M 2.6GHz processor (Sandy Bridge) running 64-bit Linux 12.01. The turbo boost is turned off, so the CPU runs at 2.6GHz in the experiment. The compiler being used is gcc 4.5.2, and the options "-O3 -msse2 -maes -mavx" 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 and 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 [27] and their website [26]. The performance comparison is given in Table 3. For 4096-byte messages, the speed of AES-JAMBU is about 11.6 cpb. Since in AES-JAMBU, each AES encryption is used to process only 8 byte of message, we expect that the speed of AES-JAMBU will be about two times slower than AES-GCM. The speed turns out to be about four times slower in our experiment. The reason may be largely due to some operations in AES-JAMBU are not optimized in our current implementation.

Table 4: The software speed comparison (in cycles per byte) for different message length on Intel Sandy Bridge. A plus sign (+) indicates that the data are from the ALE designers and the performance is measured on Intel i5-2400 microprocessor.

	64B	128B	256B	512B	1024B	4096B
AES-128-CTR$^+$	–	1.61	1.22	0.99	0.87	0.77
AES-128-CCM	7.26	6.31	5.65	5.19	5.17	5.05
AES-128-GCM$^+$	–	4.95	3.88	3.33	3.05	2.90
AES-128-OCB3$^+$	–	2.69	1.79	1.34	1.12	0.88
AES-JAMBU	17.7	14.54	13.05	12.27	11.86	11.60

We also test the JAMBU mode for SIMON64/128, SIMON96/128 and SIMON128/128, the speeds are 61 cpb, 58 cpb and 50 cpb respectively.

10 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 for 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.

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.

11 Conclusion

We have described the lightweight authenticated encryption mode JAMBU. This mode only adds a small register with size half of the block size and achieves the birthday bound security in message authentication under the nonce-misuse scenario. JAMBU is currently the most lightweight authentication encryption mode. Besides, we proposed an authenticated cipher AES-JAMBU is constructed using AES-128 as the underlying block cipher in JAMBU mode.

References

1. E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, and K. Yasuda. AES-COPA v.1. Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
2. 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.
3. M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In *Fast Software Encryption*, pages 389–407. Springer, 2004.
4. 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.
5. A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer Berlin Heidelberg, 2007.
6. A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser. ALE: AES-Based Lightweight Authenticated Encryption. In *Fast Software Encryption*, 2013.
7. J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. Thomsen, and T. Yalçn. PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications. In X. Wang and K. Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer Berlin Heidelberg, 2012.
8. CAESAR. Competition for Authenticated Encryption: Security, Applicability, and Robustness. `http://competitions.cr.yp.to/caesar.html`.
9. N. Datta and M. Nandi. ELmD v1.0. Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
10. C. De Canniere, O. Dunkelman, and M. Knežević. KATAN and KTANTANa family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 272–288. Springer, 2009.

11. I. Dinur and J. Jean. Cryptanalysis of FIDES. In *Fast Software Encryption–FSE 2014*. Springer, 2014.
12. 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.
13. 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.
14. V. Grosso, G. Leurent, F.-X. Standaert, K. Varici, F. Durvaux, L. Gaspar, and S. Kerckhof. SCREAM & iSCREAM. Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
15. J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw. The LED Block Cipher. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems–CHES 2011*, volume 6917 of *LNCS*, pages 326–341. Springer Berlin Heidelberg, 2011.
16. W. Hongjun and H. Tao. JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU (v1). Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
17. H. Hosseini and S. Khazaei. CBA Mode (v1). Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
18. ISO/IEC 19772:2009. *Information technology – Security techniques – Authenticated encryption.* ISO, Geneva, Switzerland, 2009.
19. T. Iwata, K. M. aned Jian Guo, and S. Morioka. CLOC: Compact Low-Overhead CFB v1. Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
20. T. Iwata, K. Minematsu, G. Jian, S. Morioka, and E. Kobayashi. SILC: SImple lightweight CFB. *Submission to CAESAR*, 2014.
21. 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.
22. 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.
23. C. S. Jutla. Encryption Modes with Almost Free Message Integrity. In *Advances in Cryptology – EUROCRYPT 2001*, pages 529–544. Springer, 2001.
24. 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.
25. 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.
26. 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/`.
27. T. Krovetz and P. Rogaway. The Software Performance of Authenticated-Encryption Modes. In *Fast Software Encryption – FSE 2011*, pages 306–327. Springer, 2011.
28. D. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM). `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf`.
29. K. Minematsu. AES-OTR v1. Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.

30. M. Montes and D. Penazzi. AES-CPFB v1. Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
31. NIST. Recommendation for Block Cipher Modes of Operation-Methods and Techniques. NIST special publication 800–38A, 2001 Edition.
32. T. Peyrin, S. M. Sim, L. Wang, and G. Zhang. Cryptanalysis of JAMBU. *Fast Software Encryption – FSE 2015*, 2015.
33. 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.
34. 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.
35. L. Wang. SHELL v1. Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
36. 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.
37. 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.
38. L. Zhang, W. Wu, H. Sui, and P. Wang. iFeed[AES] v1. Submission to CAESAR. Available from: `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.