Cryptanalysis of JAMBU

Thomas Peyrin[1] Siang Meng Sim[1] Lei Wang[1]
Guoyan Zhang[1,2,3]

1.Nanyang Technological University, Singapore

2.School of Computer Science and Technology, Shandong University, China

3.Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, China

10 March 2015

Table of Contents

Table of Contents

CAESAR Candidate: JAMBU

Designers: Hongjun WU, Tao HUANG (NTU, Singapore)

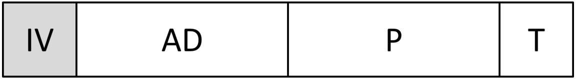
- mode of operation is similar to OFB
- 2n-bit block cipher as underlying cipher
- process blocks of n-bit information

AES–JAMBU: parameters

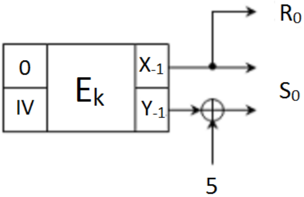
AES–JAMBU is JAMBU with AES-128 as the underlying cipher:

- associated data + plaintext $< 2^{64}$ bits under the same key
- key $=$ 128 bits
- tag $=$ 64 bits
- Initialization Vector/Nonce $=$ 64 bits

AES-JAMBU: initialisation

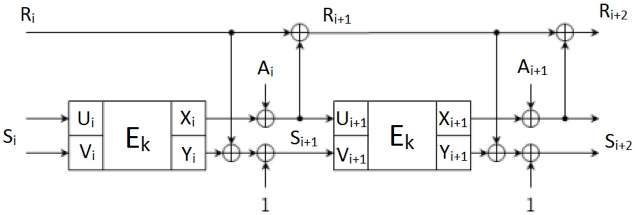


Initial input: 64-bit zeroes and 64-bit nonce (IV)



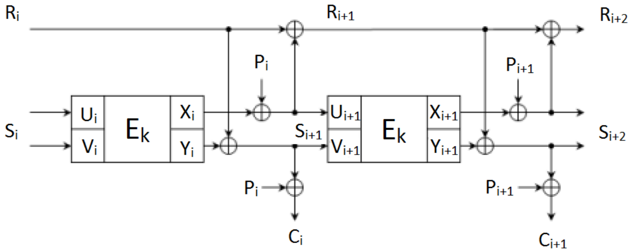
AES-JAMBU: processing of associated data

Associated data A is split into 64-bit blocks A_i

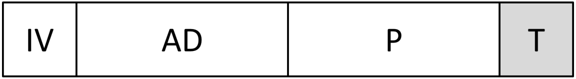


AES-JAMBU: processing of plaintext

Plaintext P is split into 64-bit blocks P_i
Ciphertext C is split into 64-bit blocks C_i



AES-JAMBU: tag generation



Last block P_M is padded with $1\|0^*$ and output is truncated. If last block is a full block, an additional block of $1\|0^{63}$ is processed without output.

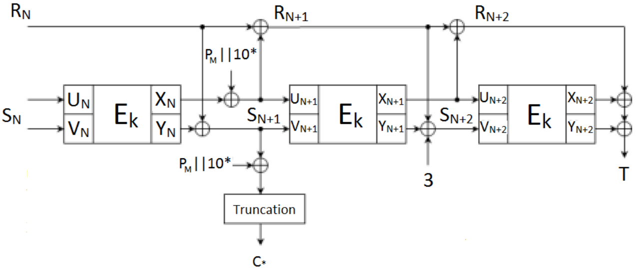


Table of Contents

JAMBU: hardware performance

JAMBU is a hardware-oriented candidate:

compared with other AE modes instantiated with a $2n$-bit block cipher, JAMBU minimizes state size, which is an advantage for hardware implementations.

Modes	State size
GCM	$6n$
OCB3	$6n$
EAX	$8n$
JAMBU	$3n$

JAMBU: software performance

On an Intel Core i5-2540M 2.6GHz processor with AES-NI:

	512-byte messages
AES-128-CCM	5.19 c/B
AES-128-GCM	3.33 c/B
AES-128-OCB3	1.34 c/B
AES−JAMBU	12.27 c/B

According to the designers, AES−JAMBU should be about two times slower than AES-GCM (their implementation is not optimized yet).

JAMBU: security claims

	confidentiality (bits)	**integrity** (bits)
nonce-respecting	128	64
nonce-misuse	128*	not specified

*: except for first block or common prefix of the message.

The designers gave very good arguments why a successful forgery should require 2^{64} computations.

"In case that the IV is reused under the same key, the confidentiality of AES-JAMBU is only partially compromised as it only leaks the information of the first block or the common prefix of the message. And the integrity of AES-JAMBU will be less secure but not completely compromised."

JAMBU: security claims

	confidentiality (bits)	**integrity** (bits)
nonce-respecting	128	64
nonce-misuse	128*	not specified

*: except for first block or common prefix of the message.

Our attack:

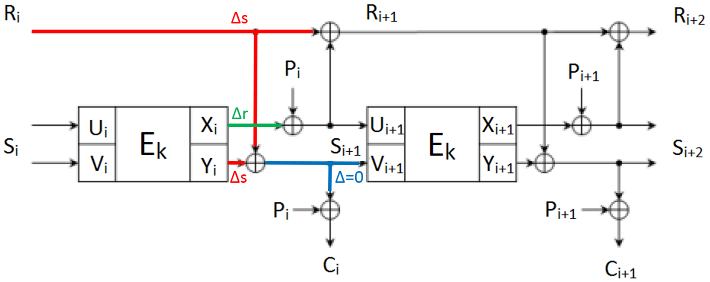
with about 2^{34} queries and computations, we can produce a valid ciphertext block corresponding to some plaintext with a prefix that has never been queried before.

Table of Contents

Table of Contents

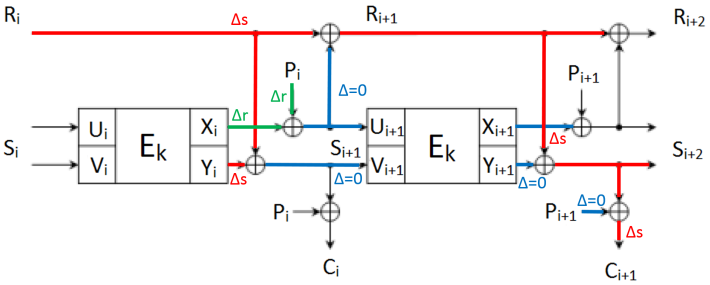
Observation 1

- no difference in V_{i+1}
  $\Rightarrow$ the differences in R_i and Y_i are the same Δs
- let the difference in X_i be Δr



Observation 2

- if the input difference in P_i is equal to Δr
  $\Rightarrow$ the difference in U_{i+1} will be cancelled out, and with no difference in P_{i+1}
  $\Rightarrow$ the output difference in C_{i+1} will be Δs



Attack Overview

Objective

Find such a diff. structure, and find the values of Δr and Δs.

Problem

Seems hard to achieve: naively building the structure costs 2^{64} computations, and we have no way of checking if we indeed found it (Δs is unknown).

Solution

"Divide-and-conquer"

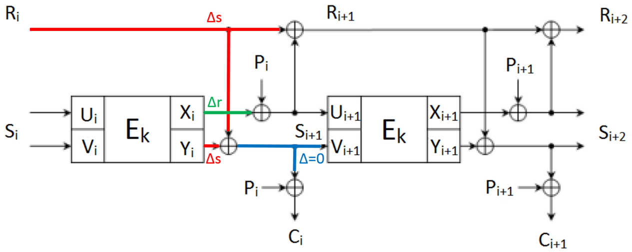
- use birthday attack to find a pair of nonce values partially follows this differential structure (nonce-respecting)
- enumerate all possible input differences in the plaintext block to force the rest of the differential structure and to find Δr and Δs (nonce-misuse)

Table of Contents

Step 1: birthday attack on V_{i+1}

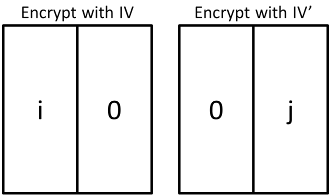
Using birthday attack, a collision on V_{i+1} can be found with about 2^{32} encryption queries:

- query for encryption for the same one block of plaintext P_1 with 2^{32} difference nonce IV
- find a collision in the ciphertext $C_1 = C'_1$
- store the pair of nonce values IV and IV'



Step 2: finding Δr and Δs

To enumerate all 2^{64} possible input differences of P_i, we use 2 sets of 2^{32} plaintext blocks.

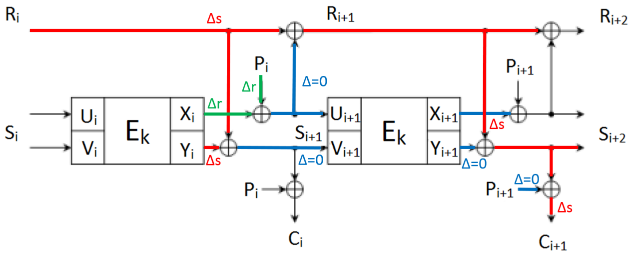


i and j ranged from 0 to $2^{32} - 1$

Any possible input difference $[i \| j]$ can be formed with a pair of plaintext blocks $[i \| 0^{32}]$ and $[0^{32} \| j]$.

Step 2: finding Δr and Δs

P_{i+1} is set to a constant value (e.g. all zeros)

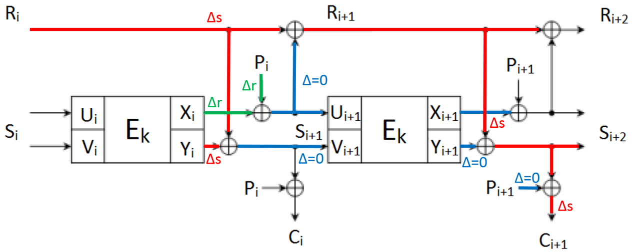


We ask for the encryption of $[i \| 0^{32}] \| [0^{64}]$ with nonce IV and $[0^{32} \| j] \| [0^{64}]$ with nonce IV'.

Step 2: finding Δr and Δs

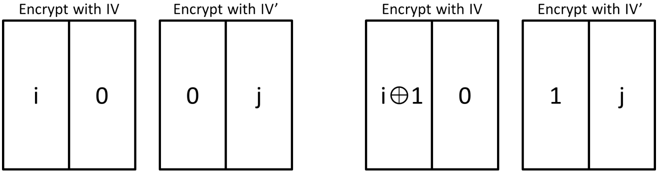
Question: how do we know that we insert the right Δr in P_i?

Answer: the right Δr will give the same output difference Δs in the second block independent of the plaintext value in the first block.



Step 2: finding Δr and Δs

The right Δr will give the same output difference Δs independent of the value of P_i, so we build a few tables.



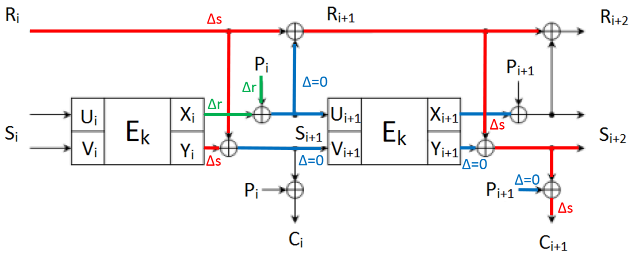
i and j ranged from 0 to $2^{32}-1$

If $\Delta r = [i \| j]$, then $C_2[i \| 0] \oplus C_2[0 \| j] = C_2[i \oplus 1 \| 0] \oplus C_2[1 \| j] = \Delta s$.

Note that first and third tables are the same up to permutation.
Hence, we need $3 \cdot 2^{32}$ encryption queries.

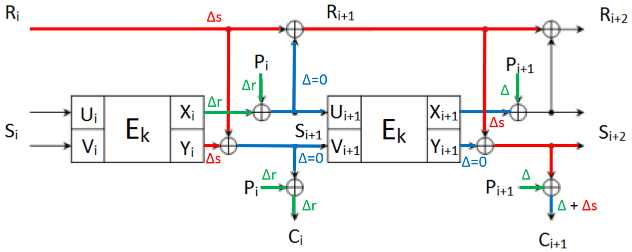
Step 2: summary

- query for $3 \cdot 2^{32}$ encryptions
- compute and store the difference of the second block of the ciphertexts
- find the collision
  $C_2[i\|0] \oplus C_2[0\|j] = C_2[i \oplus 1\|0] \oplus C_2[1\|j] = \Delta s$.
- obtain $\Delta r = [i\|j]$ and $\Delta s = C_2[i\|0] \oplus C_2[0\|j]$.



Step 3: forging a valid ciphertext block

For any choice of plaintext blocks P_1, P_2, by querying $[P_1 \oplus \Delta r] \| [P_2 \oplus \Delta]$ with nonce IV and obtaining the ciphertext $[C_1 \| C_2]$, we can deduce the ciphertext of $[P_1 \| P_2]$ encrypted with nonce IV' to be $[C_1 \oplus \Delta r] \| [C_2 \oplus \Delta \oplus \Delta s]$, where Δ can be any difference.



Note that $[P_1]$ is a different prefix that has never been queried before.

Complexity Evaluation of the Attack

- Step 1 requires about 2^{32} queries (nonce-respecting)
- Step 2 requires $3 \cdot 2^{32}$ queries (nonce-misuse)
- Step 3 requires a single query

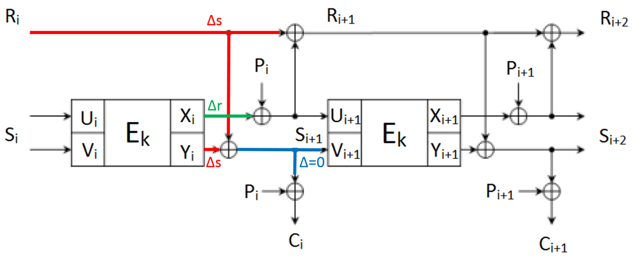
With only about 2^{34} queries, we can deduce the ciphertext corresponding to a plaintext with a prefix that has never been queried before.

Attack has been implemented and verified!

Numerical Example: Step 1

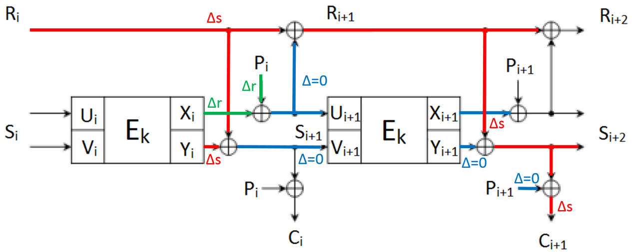
For simplicity, the associated data was set to be empty.

K :	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	10
IV :	b1	ef	89	a0	4e	21	30	bd								
IV' :	10	5a	1f	5b	34	49	1e	5c								
P_1 :	7f	95	77	ca	09	77	a8	a5								
C_1 :	2d	2b	58	18	fa	f5	af	f1								
C'_1 :	2d	2b	58	18	fa	f5	af	f1								



Numerical Example: Step 2

$[i\|0^{32}]\|[P_2]$	:	60	28	6d	74	00	00	00	00	00	00	00	00	00	00	00	00
$C_2[i\|0]$	:									af	45	56	9e	26	c6	7e	d0
$[0^{32}\|j]\|[P_2]$	:	00	00	00	00	93	47	1e	92	00	00	00	00	00	00	00	00
$C_2[0\|j]$	:									73	79	44	54	a7	b4	5b	4c
Δr	:	60	28	6d	74	93	47	1e	92								
Δs	:									dc	3c	12	ca	81	72	25	9c



Numerical Example: Step 3

We query arbitrary plaintext blocks $[P_1]\|[P_2]$ with IV and deduce the ciphertext of $[P_1 \oplus \Delta r]\|[P_2]$ with IV' as $[C_1 \oplus \Delta r]\|[C_2 \oplus \Delta s]$. Note that $[P_1 \oplus \Delta r]$ is a prefix that has never been queried before.

IV :	b1 ef 89 a0 4e 21 30 bd	
$[P_1]\|[P_2]$:	95 d9 43 9e 0b 4d 6d 27	6a ba db 0a 12 f8 13 45
$[C_1]\|[C_2]$:	c7 67 6c 4c f8 cf 6a 73	6b 05 9b c6 fc e6 7a ee
Δr :	60 28 6d 74 93 47 1e 92	
Δs :		dc 3c 12 ca 81 72 25 9c
$[C_1^D]\|[C_2^D]$:	a7 4f 01 38 6b 88 74 e1	b7 39 89 0c 7d 94 5f 72

Lastly, we verify our deduced ciphertext.

IV' :	10 5a 1f 5b 34 49 1e 5c	
$[P_1 \oplus \Delta r]\|[P_2]$:	f5 f1 2e ea 98 0a 73 b5	6a ba db 0a 12 f8 13 45
$[C_1']\|[C_2']$:	a7 4f 01 38 6b 88 74 e1	b7 39 89 0c 7d 94 5f 72

Table of Contents

Conclusion

We have shown a generic confidentiality attack on the JAMBU operating mode:

- the attack is independent of the underlying block cipher
- in the nonce-misuse scenario
- practical when instantiated with AES: only about 2^{34} queries
- attack verified by implementation

How about nonce-respecting scenario?

One can apply the same idea to break IND-CCA2 security of JAMBU in the nonce-respecting scenario:

- during Step 2 of the attack, use decryption queries in order to repeat nonces...
- ... but one has to pay 2^{64} to guess the tag and get corresponding plaintext from the oracle
- final complexity of $O(2^{32}) \times 2^{64} = O(2^{96})$ queries and computations to break IND-CCA2 security

but the security model for the security claims of JAMBU was not given by the designers (they didn't mean IND-CCA2)

Thank you. :)