Cryptanalysis of JAMBU

Thomas Peyrin
(j.w. with Siang Meng Sim and Lei Wang and Guoyan Zhang)

NTU - Singapore

ESC 2015

Clervaux, Luxembourg - January 16, 2015

NANYANG TECHNOLOGICAL UNIVERSITY

Outline

1. The JAMBU candidate
2. Performance and security claims
3. Nonce-misuse attack on JAMBU
   - Differential structure in JAMBU
   - Details of the attack
4. Conclusion

CAESAR candidate: JAMBU

Designers: Hongjun WU, Tao HUANG (NTU)

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

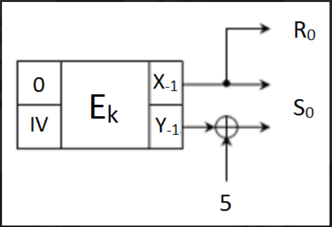
AES-JAMBU: parameters

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

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

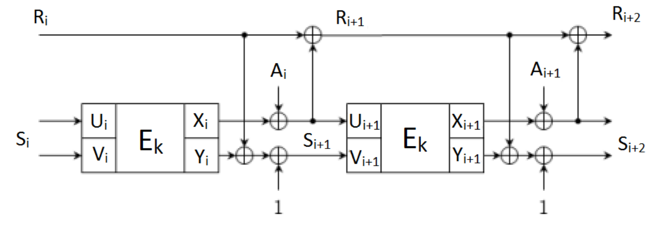
AES-JAMBU: initialisation

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



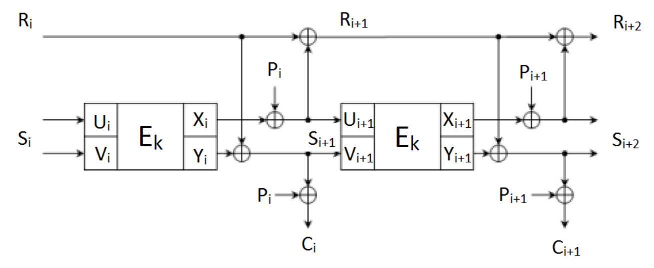
AES-JAMBU: processing of associated data

A is split into 64-bit blocks A_i



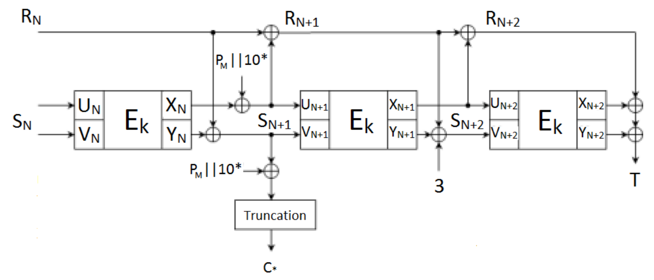
AES-JAMBU: processing of plaintext

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.



Outline

1. The JAMBU candidate
2. Performance and security claims
3. Nonce-misuse attack on JAMBU
   - Differential structure in JAMBU
   - Details of the attack
4. Conclusion

JAMBU: hardware performance

JAMBU is a hardware-oriented candidate:

compared with other AE modes instantiated with a $2n$-bit block cipher, JAMBU minimizes the 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 authors give very good arguments why a successful forgery should require 2^{64} computations

JAMBU: security claims

"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:

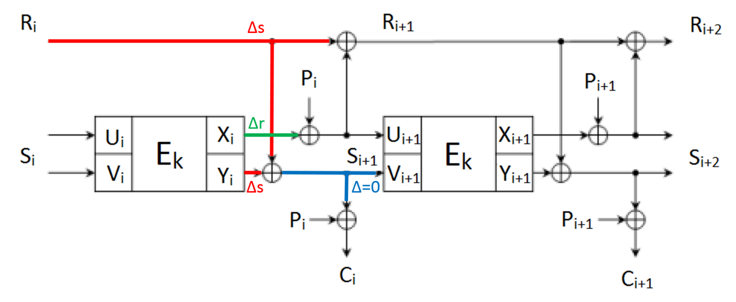
with about 2^{34} queries and computations, we can produce a valid ciphertext block corresponding to some plaintext with a prefix that has never been queried before

Outline

Outline

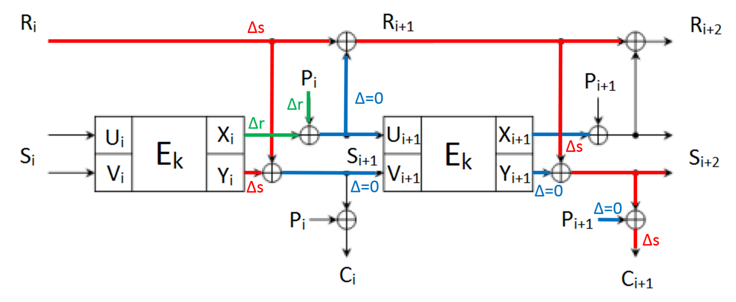
Observation 1

- 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} to be Δs



Attack overview

Objective

Build 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 secret)

Solution

"Divide-and-conquer"

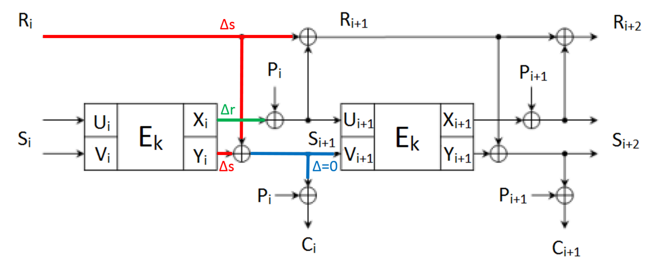
- use birthday attack to find a pair of nonce values that partially follows this differential structure (nonce-respecting)
- enumerate all possible input differences in the plaintext block to force the rest of the differential structure and to find Δr and Δs (nonce-misuse)

Outline

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 ... and we can detect it:

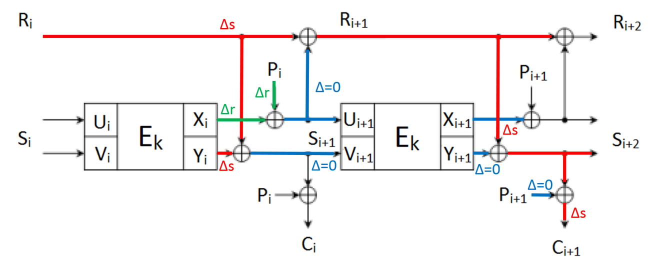
- query for encryption for the same one block of plaintext P_1 with 2^{32} difference nonce IV
- find a collision in the ciphertext $C_1 = C_1'$
- store the pair of nonce values IV and IV'



Step 2: finding Δr and Δs

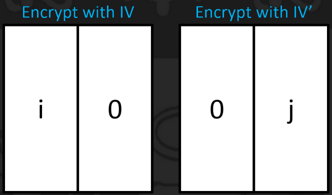
Question: How do we know that we insert the right Δr in P_i ?

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

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

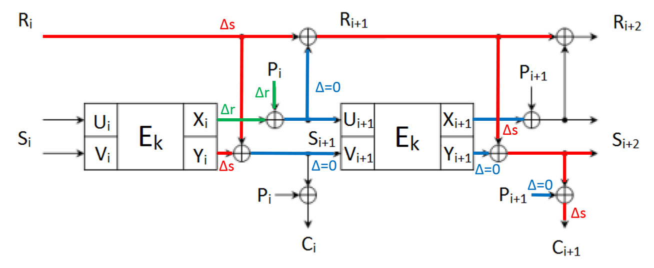


i and j ranged from 0 to $2^{32} - 1$

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

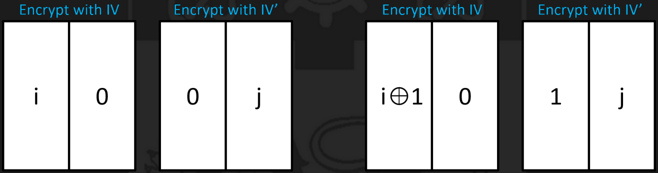
P_{i+1} is set to a constant value (i.e. all zeros)



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

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



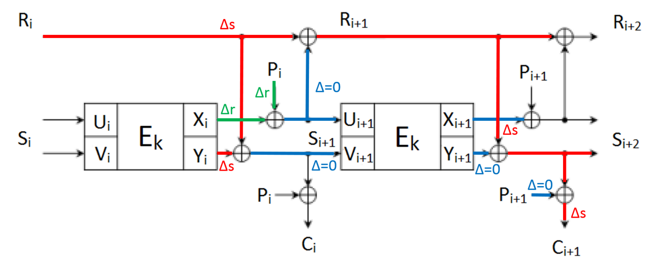
i and j range from 0 to $2^{32} - 1$

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: we need $3 \cdot 2^{32}$ encryption queries

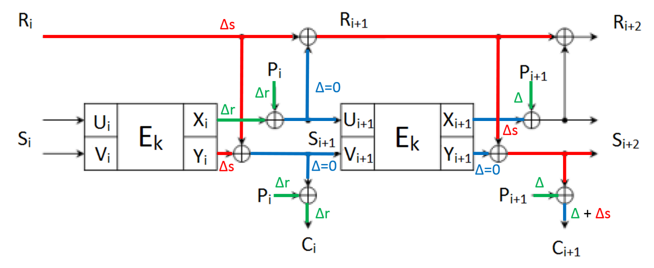
Step 2: summary

- 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 $[P_1 \oplus \Delta r] \| [P_2 \oplus \Delta]$, we can deduce the ciphertext 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 $[P_1 \oplus \Delta r]$ 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 !

Outline

Conclusion

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

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

What about nonce-respecting scenario ?

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

- apply exactly the same 2^{32} attack using decryption queries, so you can repeat nonces ...
- ... but every time you query a ciphertext, you have to pay 2^{64} to guess the tag and get the corresponding plaintext from the oracle
- final complexity of $2^{32} \times 2^{64} = 2^{96}$ 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)

Open positions @ NTU - Singapore

Guo Jian: `guojian@ntu.edu.sg`

4 postdoc positions (symmetric key - lightweight crypto)

Thomas Peyrin: `thomas.peyrin@ntu.edu.sg`

2 postdoc positions and 1 PhD position
(symmetric key - lightweight crypto - side channels)

Huaxiong Wang: `hxwang@ntu.edu.sg`

1 postdoc position (coding and lattice based crypto)

Hongjun Wu: `wujh@ntu.edu.sg`

2 postdoc positions (symmetric key - computer security)

The SYmmetric and Lightweight cryptography Lab (SYLLAB):
`www1.spms.ntu.edu.sg/~syllab/m/index.php/Home`

Thank you !