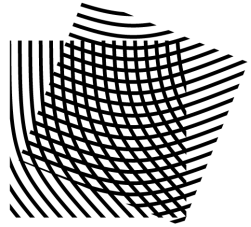
Cryptanalysis of the permutation based algorithm SpoC

Liliya Kraleva, Raluca Posteuca and Vincent Rijmen

October 19-22 2020
Lightweight Cryptography Workshop



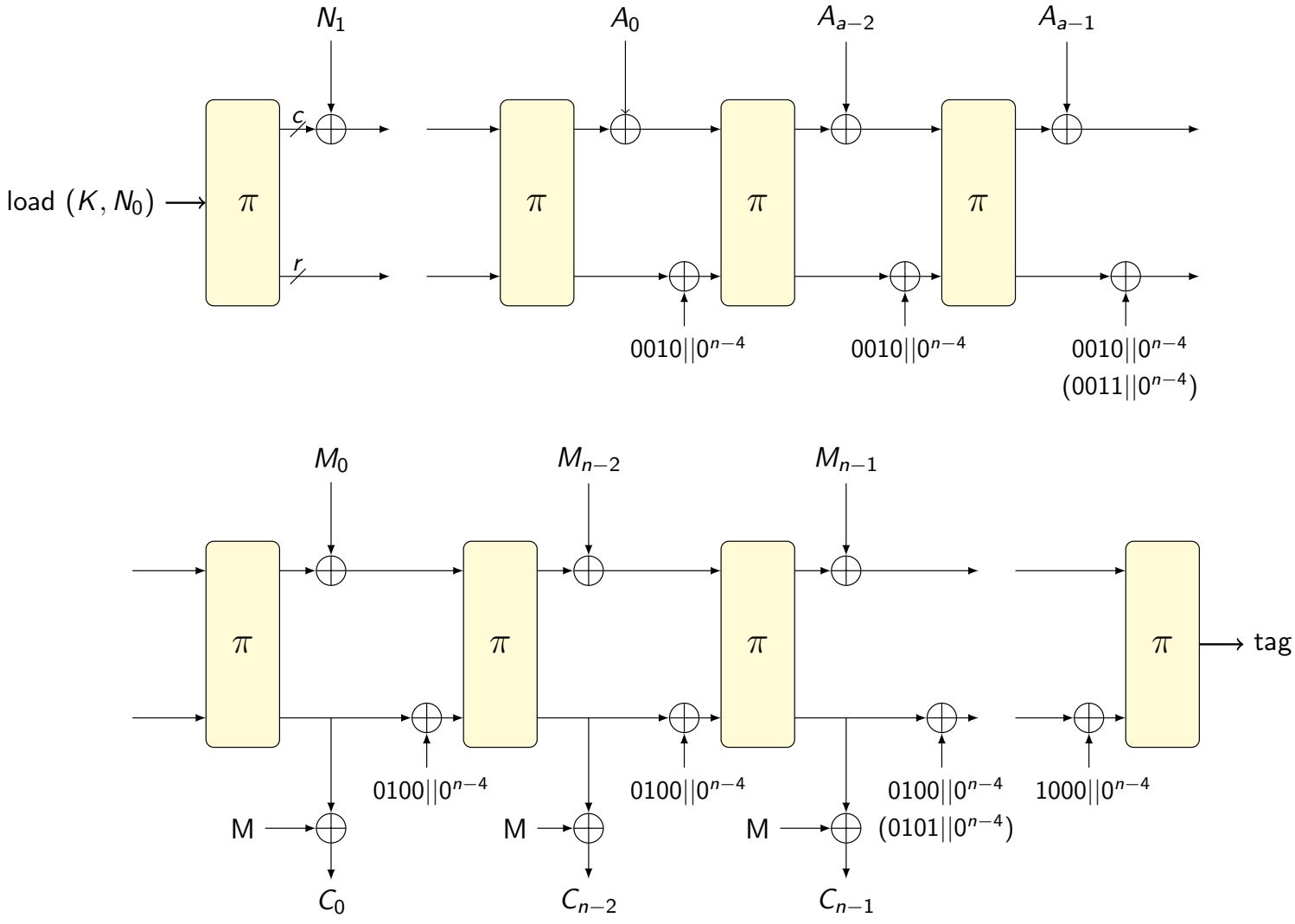
COSIC

Our contributions

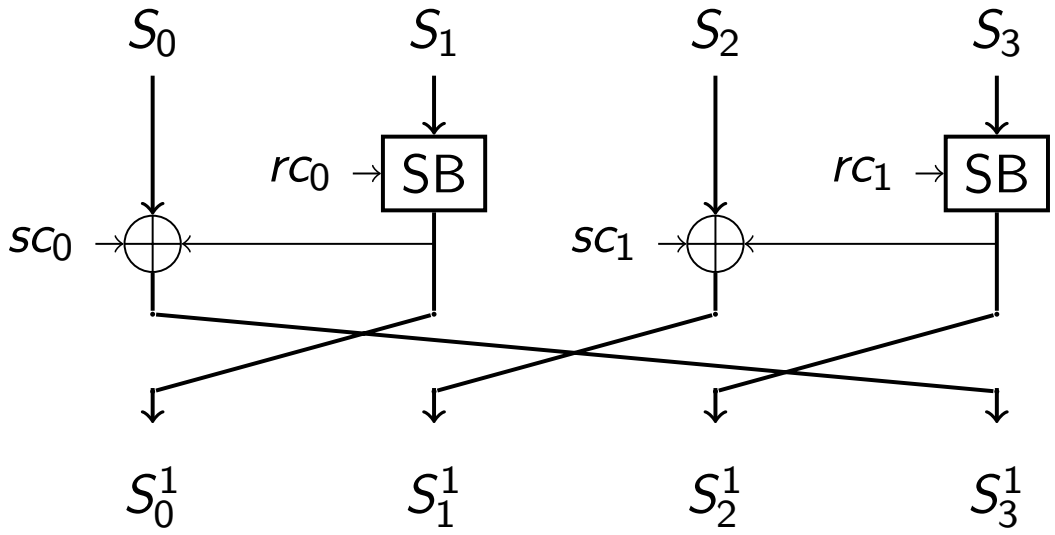
- Characteristics for sLiSCP-light-[192] and sLiSCP-light-[256] over round-reduced versions
- Tag forgery attacks on both SpoC versions based on the characteristics
- Message recovery attack based on the characteristic
- Key-recovery attack on SpoC-64, regardless of the permutation
- Observations on the constants used in sLiSCP-light.

Content

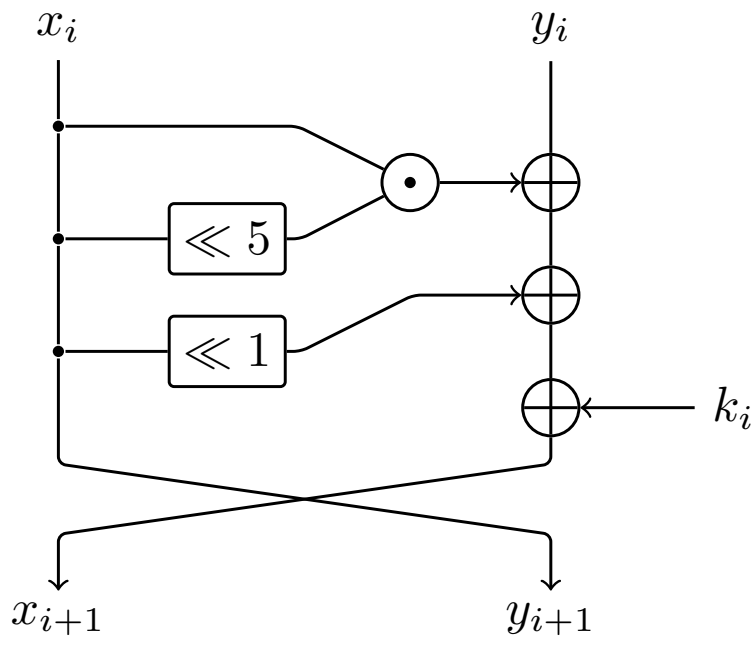
The algorithm of Spoc



sLiSCP-light



Simeck SBox



$$R(x_{i+1}, y_{i+1}) = (y_i \oplus f(x_i) \oplus rc, x_i),$$

where $f(x) = (x \quad (x \lll 5) \oplus (x \lll 1))$

Parameters

Parameters of the SpoC variants:

Instance	state	rate	key	nonce	tag
SpoC-64_sLiSCP-light-[192]	192	64	128	128	64
SpoC-128_sLiSCP-light-[256]	256	128	128	128	128

Parameters of the sLiSCP-light permutation:

permutation	state	SBox size	SBox rounds	perm. steps
sLiSCP-light-[192]	192	48	6	18
sLiSCP-light-[256]	256	64	8	18

Security claims

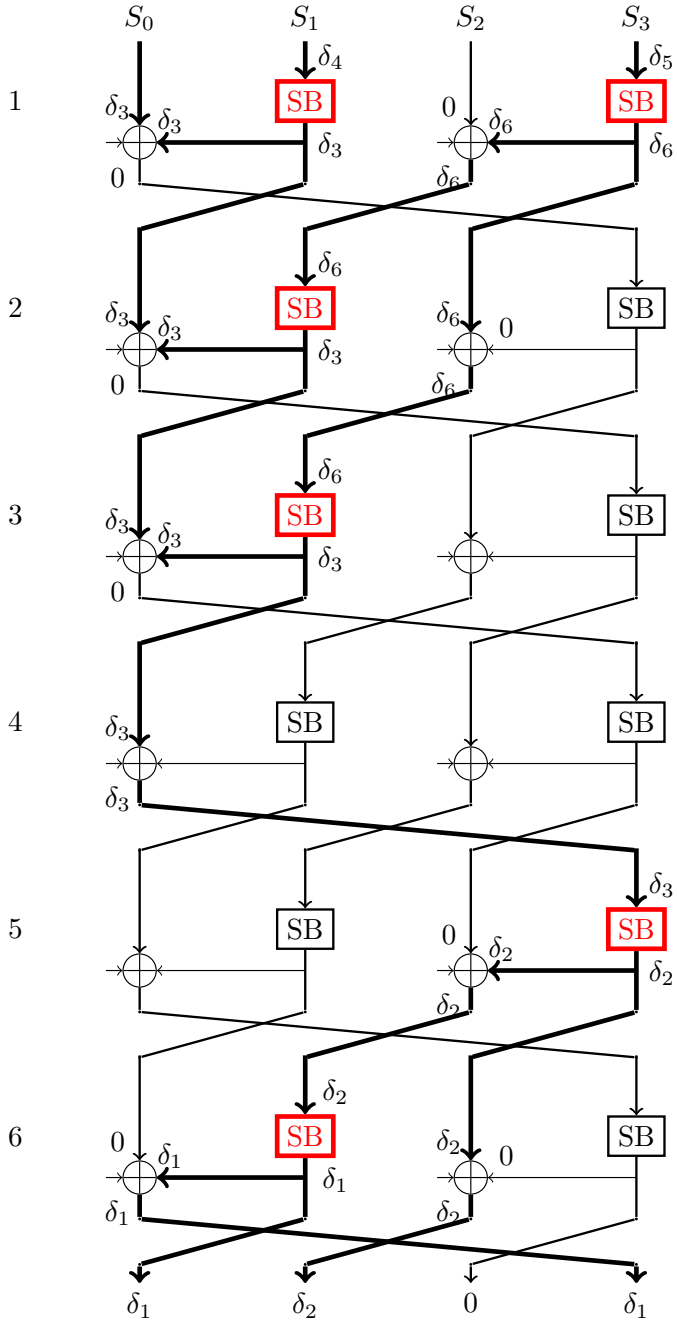
AEAD algorithm	Confidentiality		Integrity		Advantage
	Time	Data (in bytes)	Time	Data (in bytes)	
SpoC-64_sLiSCP-light[192]	2^{112}	2^{50}	2^{112}	2^{50}	2^{-16}
SpoC-128_sLiSCP-light[256]	2^{112}	2^{50}	2^{112}	2^{50}	2^{-16}

- Our attacks $\rightarrow$ more than 2^{50} data?

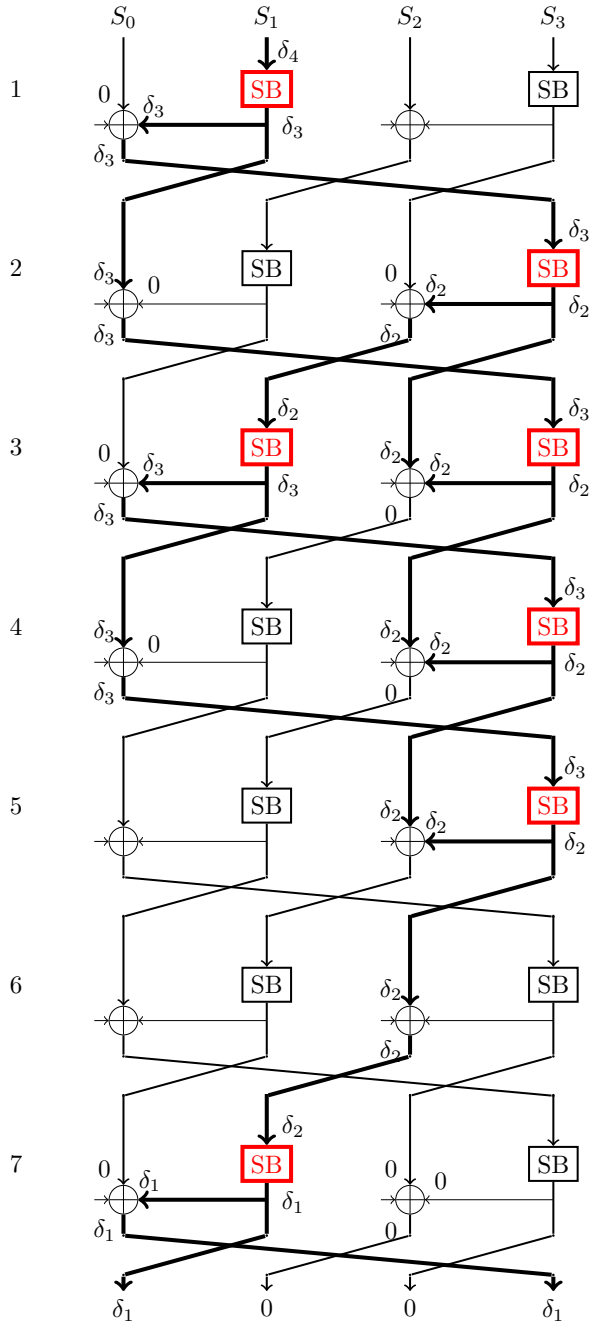
Content

Differential Characteristics of sLiSCP-light

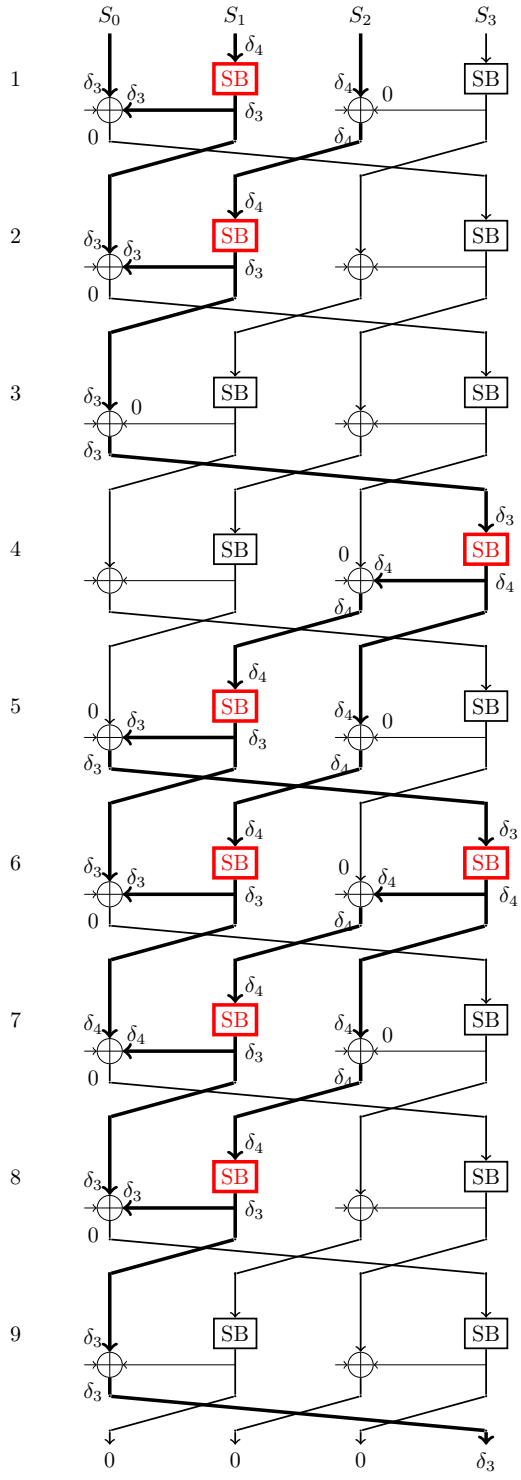
- Cover round-reduced versions
- We impose some constraints needed for the attack on SpoC
- These characteristics are not the optimal ones in general, but the best ones for our attacks



- sLiSCP-light-[256]
- 6 round (out of 18)
- constraint on the output difference
- input difference: $\delta_3||\delta_4||0||\delta_5$
- output difference: $\delta_1||\delta_2||0||\delta_1$
- best probability: $2^{-106.14}$.



- sLiSCP-light-[192]
- 7 rounds (out of 18)
- constraint on the input and output difference
- input difference: $0||\delta_4||0||0$
- output difference: $\delta_1||0||0||\delta_1$
- best probability: $2^{-108.2}$.



- sLiSCP-light-[192]
- 9 rounds (out of 18)
- constraint on the output difference
- input difference: $\delta_3||\delta_4||\delta_4||0$
- output difference: $0||0||0||\delta_3$
- best probability: $2^{-109.84}$.

Content

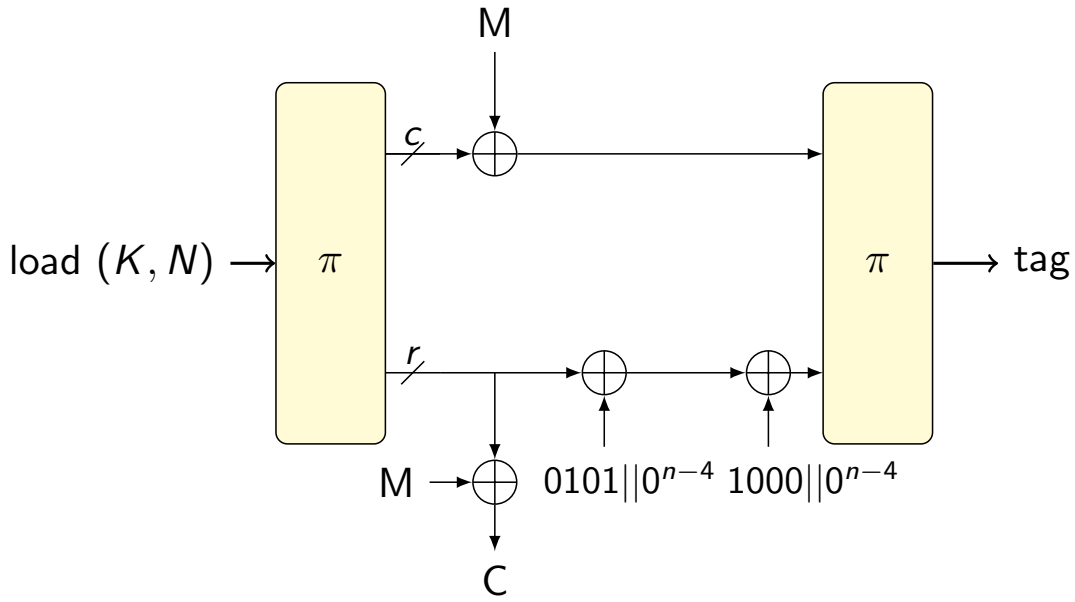
Tag forgery attacks

Based on the following observations:

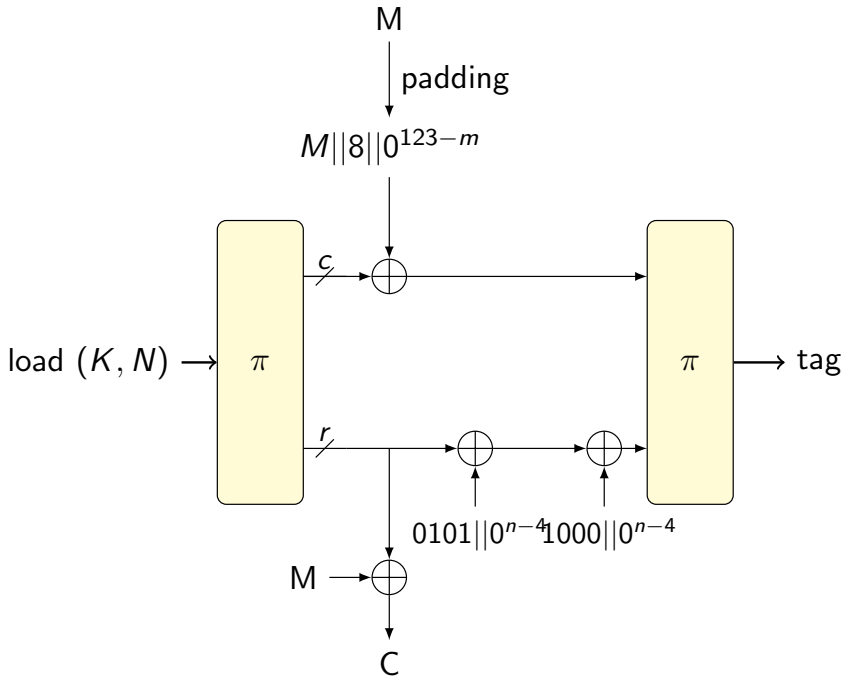
- A null AD or an empty message impose the corresponding phase to be skipped
- In each phase and depending on the block length, a different constant is added to the rate part
- Compared to SpoC-64, the initialization phase of SpoC-128 consists only of loading the key and nonce to the state.

Tag forgery attacks

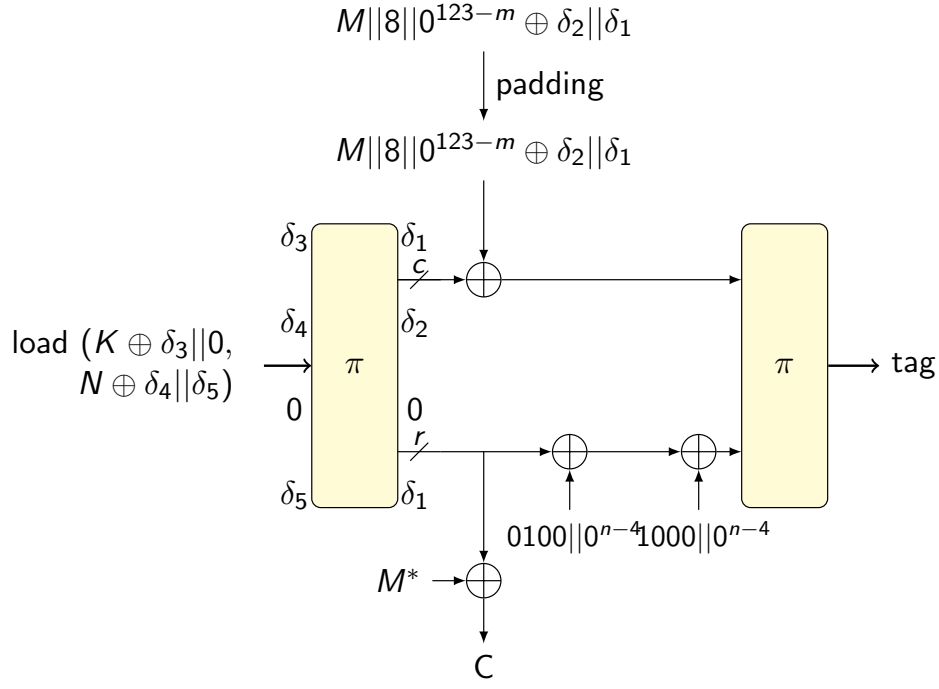
- Assume null AD, an incomplete block M and $M^* = Padded(M)$. The only difference in the processing phases is the control signals
- After π :
  The difference in the **rate bits** can be canceled by the difference of the control signals;
  The difference in the **capacity bits** can be canceled by the difference of the message blocks



Processing of messages with difference



(a)



(b)

Scenarios

Depending on the scenario, we identified three possible values for the control signals' difference, as follows:

1. $0001||0^{n-4} = 0100||0^{n-4} \oplus 0101||0^{n-4}$
   when we encrypt $(\text{""}, M)$ and $(\text{""}, M^*)$

2. $0110||0^{n-4} = 0100||0^{n-4} \oplus 0010||0^{n-4}$
   when we encrypt $(\text{""}, M)$ and $(M, \text{""})$. It will produce no ciphertext, however the tags of the two would be the same. Hence we can forge the verification of associated data.

3. $0111|0^{n-4} = 0101||0^{n-4} \oplus 0010||0^{n-4}$
   when we encrypt $(\text{""}, M)$ and $(AD, \text{""})$, where M is incomplete block and $AD = padded(M)$.

The tag-forgery attack on SpoC-128

1. With a key-nonce pair (K, N) ask for the encryption of $(\text{""}, M)$ for some block of plaintext M with length $m < 128$; obtain the ciphertext-tag pair $(C = C_1 C_2, \tau)$;
2. Verify whether the differential in the last SB holds
   - 2.1 If the condition holds, ask for the decryption of $(C_1 \oplus \delta_2 \oplus \delta_1 || C_2 \oplus \delta_1, \tau)$ under $(K \oplus \Delta_K, N \oplus \Delta_N) = (K \oplus \delta_4 || \delta_5, N \oplus \delta_3 || 0)$;
   - 2.2 If the condition does not hold, change N and/or K and repeat from step 1.

Complexity

	steps	data	time
SpoC-128	6	$2^{106.14}$	$2^{107.14}$
SpoC-64	7	$2^{108.2}$	$2^{109.2}$

- Data complexity: number of encryptions/decryptions
- Time complexity: number of Sbox calls
- Improved by having multiple characteristics
- Time-memory trade-off by generating a table with the "good" X_0

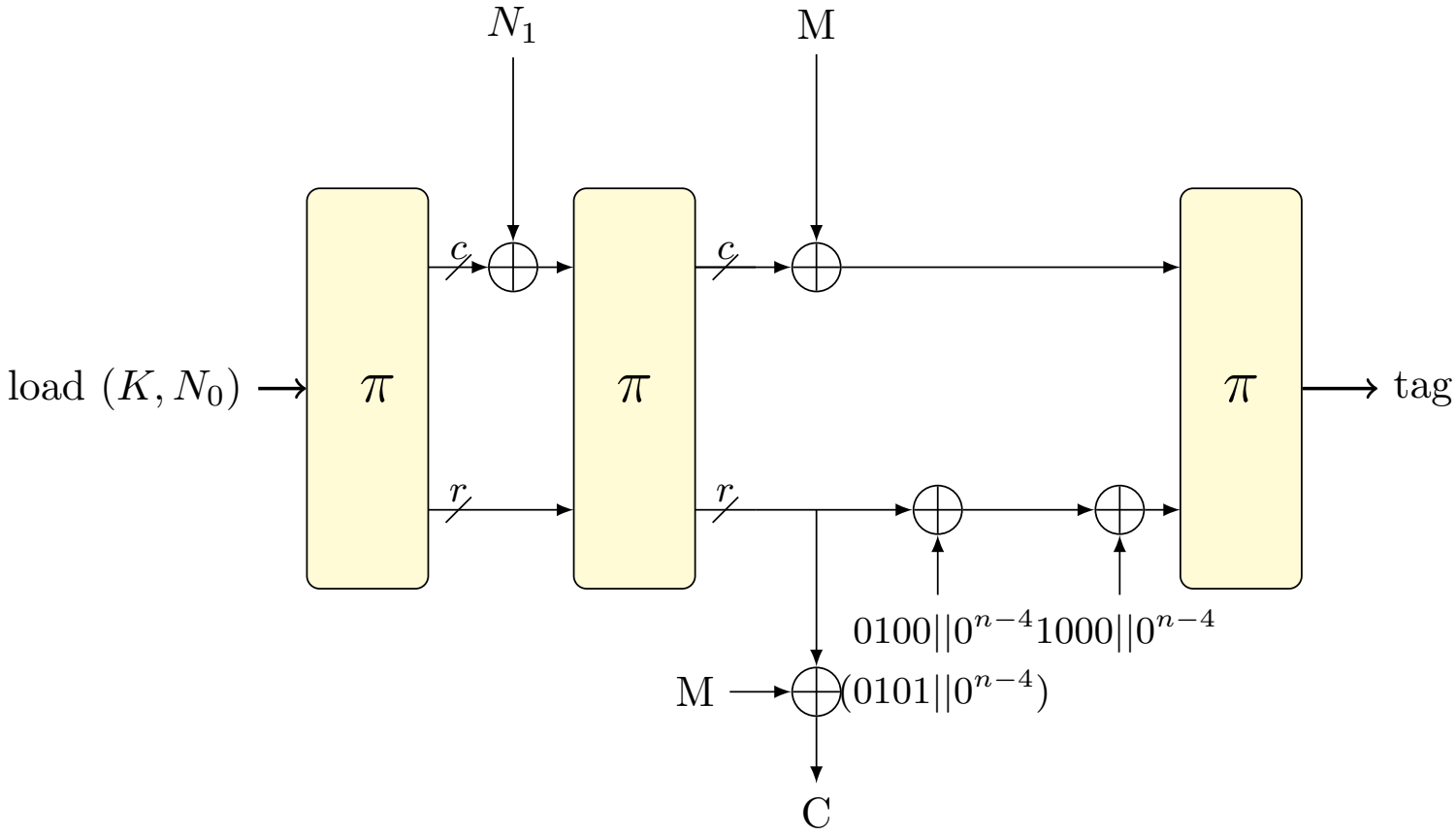
Content

Message and key-recovery attacks

- For SpoC-64, the initialization phase is not bijective; multiple (key,nonce) pairs lead to the same internal state
- We aim for a collision after the initialization phase.
- 2 ways to have a collision - differential characteristic and preimage approach
- The same message will have the same ciphertext over different (key, nonce) pairs.

Message recovery attack

- 9 rounds characteristic
- After π in the initialization phase, the difference is only in the capacity bits
- Cancelled by the addition of N_1



Message recovery attack

The attack:

1. With a key-nonce pair (K, N) ask for the encryption of an arbitrary, unknown plaintext M, using the associated data AD; we obtain the ciphertext-tag pair (C, τ);

2. Ask for the decryption of (C, τ) under $(K \oplus \Delta_K, N \oplus \Delta_N) = (K \oplus \delta_4||0, N \oplus \delta_3||\delta_4)$ and using the initial AD;

3. If the tag verification holds, we obtain the plaintext M'. If M' is a readable text, then $M' = M$ and the message is recovered

Message recovery attack

The attack:

1. With a key-nonce pair (K, N) ask for the encryption of an arbitrary, unknown plaintext M, using the associated data AD; we obtain the ciphertext-tag pair (C, τ);

2. Ask for the decryption of (C, τ) under $(K \oplus \Delta_K, N \oplus \Delta_N) = (K \oplus \delta_4||0, N \oplus \delta_3||\delta_4)$ and using the initial AD;

3. If the tag verification holds, we obtain the plaintext M'. If M' is a readable text, then $M' = M$ and the message is recovered

Data complexity: $2 \cdot 2^{109.84}$ (number of encryptions/decryptions)

Key-recovery attack

- **Def.**The (key, nonce) pairs (K_1, N_1) and (K_2, N_2) are said to be in the same *equivalence class* (or simply equivalent) if the corresponding internal states, after the initialization phase, are equal.
- 2^{192} equivalence classes
- 2^{64} (key,nonce) pairs in each class
- encrypting/decrypting the same plaintext/ciphertext with equivalent (key,nonce) pairs leads to the same ciphertext/plaintext.

Key-recovery attack

Consist of two phases:

1. **Offline phase:**
   - The adversary generates a table containing 2^{110} entries.
   - Each entry contains a $(K, N_0||N_1)$ pair and the ciphertexts and tag obtained by applying SpoC-64 on a well chosen plaintext M, under the $(K, N_0||N_1)$ pair and a null AD.
   - The (key, nonce) pairs are generated such that they belong to different equivalence classes.

Key-recovery attack

Consist of two phases:

1. **Offline phase:**
   - The adversary generates a table containing 2^{110} entries.
   - Each entry contains a $(K, N_0||N_1)$ pair and the ciphertexts and tag obtained by applying SpoC-64 on a well chosen plaintext M, under the $(K, N_0||N_1)$ pair and a null AD.
   - The (key, nonce) pairs are generated such that they belong to different equivalence classes.
2. **Online phase:**
   - Intercept random messages, encrypted by a valid user
   - The adversary verifies if the first 3 blocks of the ciphertext belong to the table
   - When a match is found, the adversary knows the internal state after the initialization
   - On the obtained internal state, the adversary XORs the N_1, applies the inverse of the permutation and recovers the key.

Complexity of the key-recovery attack

phase	data	time	memory
Offline		2^{110} enc.	2^{110} table entries
Online	2^{67}	2^{67} look ups	
Total	2^{67}	2^{110}	2^{110} table entries

Then the probability of success is 2^{-15}, twice as the authors claim.

Content

Observations on the constants

$$rc_0^i = rc_1^{i+8}, \forall i \in \{0, ...10\},$$

$$sc_0^i = sc_1^{i+8}, \forall i \in \{0, ...10\}$$

step i	(rc_0^i, rc_1^i)
0-5	(f, 47), (4, b2), (43, b5), (f1, 37), (44, 96), (73, ee)
6-11	(e5, 4c), (b, f5), (47, 7), (b2, 82), (b5, a1), (37, 78)
12-17	(96, a2), (ee, b9), (4c, f2), (f5, 85), (7, 23), (82, d9)

step i	(sc_0^i, sc_1^i)
0-5	(8, 64), (86, 6b), (e2, 6f), (89, 2c), (e6, dd), (ca, 99)
6-11	(17, ea), (8e, 0f), (64, 04), (6b, 43), (6f, f1), (2c, 44)
12-17	(dd, 73), (99, e5), (ea, 0b), (0f, 47), (04, b2), (43, b5)

Conclusion

Attack	rounds of π	data	time	memory
Tag forgery on SpoC-128	6	$2^{106.14}$	$2^{107.14*}$	-
Tag forgery on SpoC-64	7	$2^{108.2}$	$2^{109.2*}$	-
Message recovery on SpoC-64	9	$2^{110.84}$	$2^{109.84**}$	-
Key recovery on SpoC-64	all	2^{67}	2^{110}	2^{110***}

* SBox computations
** table look ups
*** table entries

Table: All attacks on SpoC and their complexities.

Thank you for your attention!

Liliya.Kraleva@esat.kuleuven.be
Raluca.Posteuca@esat.kuleuven.be

Observations on the constants

- The constants are computed using an LFSR with length 7 and the primitive polynomial $x^7 + x + 1$. The initial state of the LFSR is filled with seven bits of 1.
- The LFSR runs continuously for $288 = 18 \times 2 \times 8$ steps. The first 16 bits of the returning string are: 1111111000000100.
- The bits of rc_0^0 are the bits in odd positions of the string above while the bits of rc_1^0 are the bits from the even positions, both of them being read in an little-endian manner. Thus, $rc_0^0 = 00001111 = 0xF$ and $rc_1^0 = 01000111 = 0x47$.
- Since the primitive polynomial has degree 7, it's period is $2^7 - 1 = 127$. Therefore, the $127 + n^{th}$ bit will be equal to the n^{th} generated bit.
- In particular, the bits of rc_1^{8+n} are equal to the bits of rc_0^n.