Review of the White-Box Encodability of NIST Lightweight Finalists

Alex Charlès[1] and Chloé Gravouil[2]

1. Université de Rennes 1
2. EDSI - KUDELSKI Group

NIST Lightweight Cryptography Workshop 2022

Outline

- Limitations of encodings for white-box implementations
- Presentation of our encoding solution that avoids these limitations
- Review of the white-box encodability of the NIST LWC finalists
- Presentation of our solution applied to GIFT

UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Limitations of encodings for white-box implementations

UNIVERSITÉ DE RENNES 1

NAGRA KUDELSKI

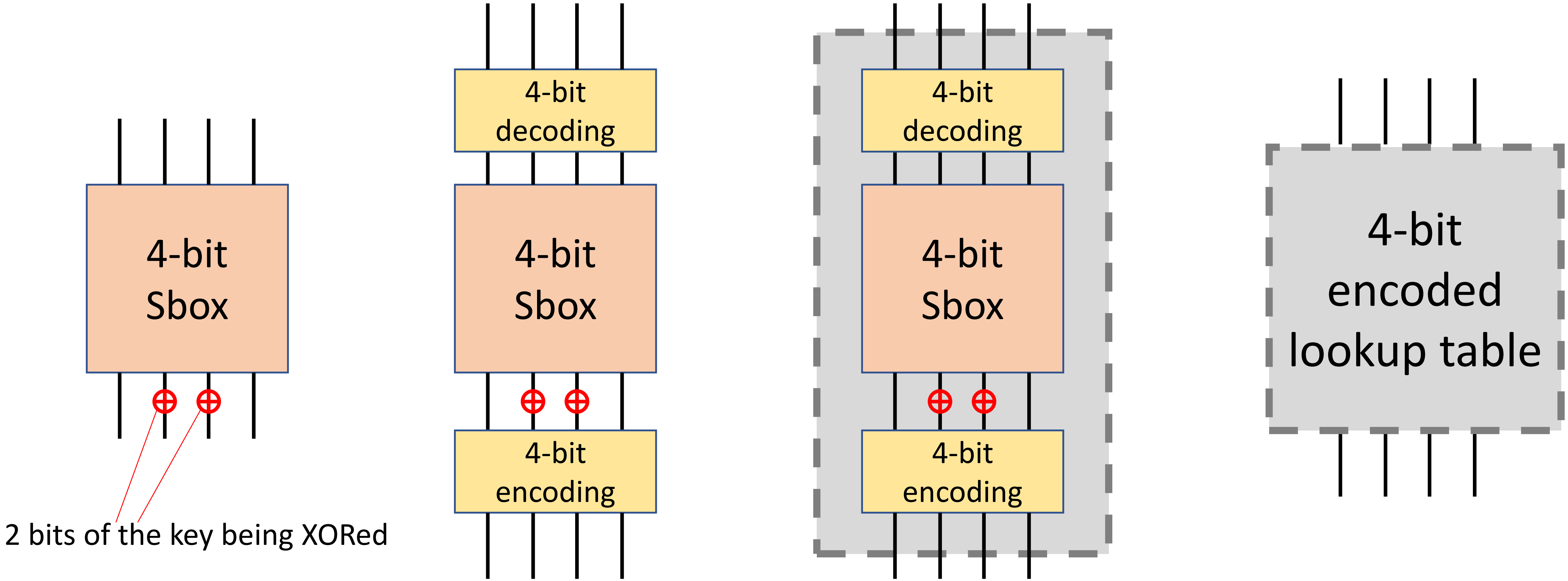
Quick overview of white-box cryptography

- A white-box adversary has full access to a software implementation and its execution platform and wants to extract key information

- A method first proposed by Chow *et al.* to protect a constant key is to tabularize the operations with encodings

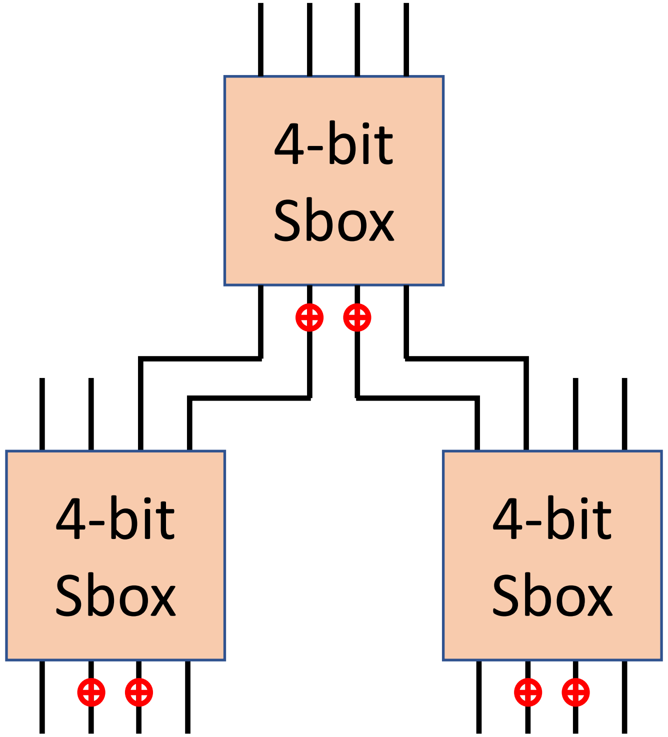
UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

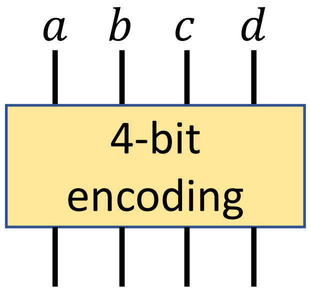
Protecting key bits XORed with encoding



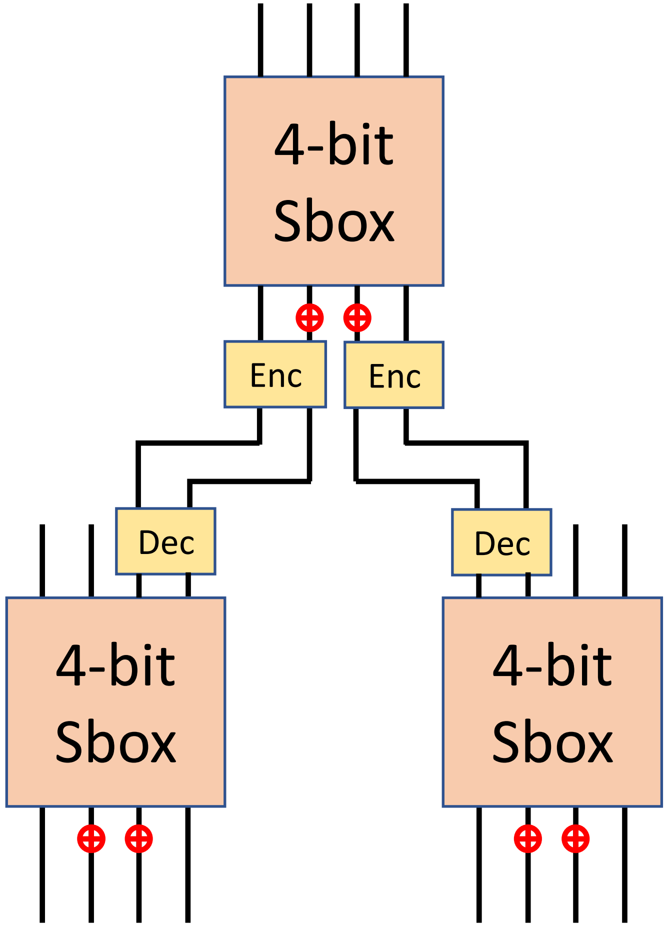
UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

We cannot always encode all the output bits together





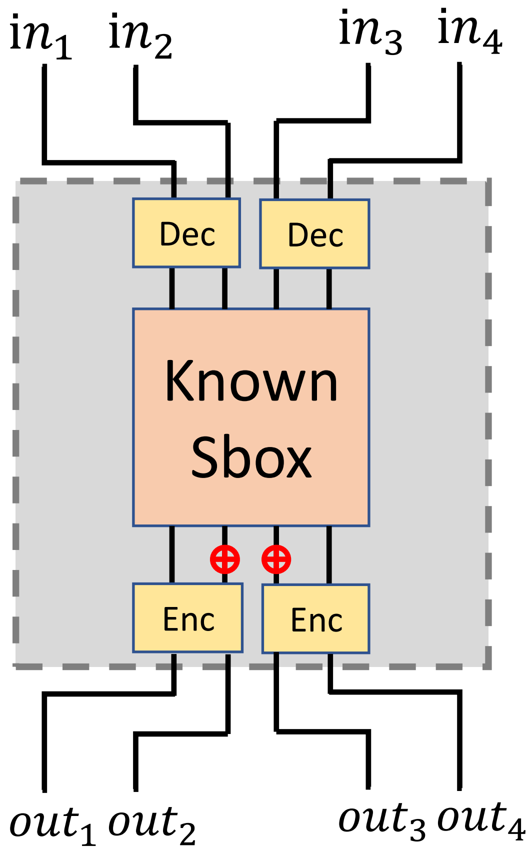
In order to recover the bit a from this 4-bit encoding, we need all its output bits.



Small encoding are weak to brute-force and differential attacks

- Brute-force attack : If we extend the previous example, an attacker has 2^{20} possibilities

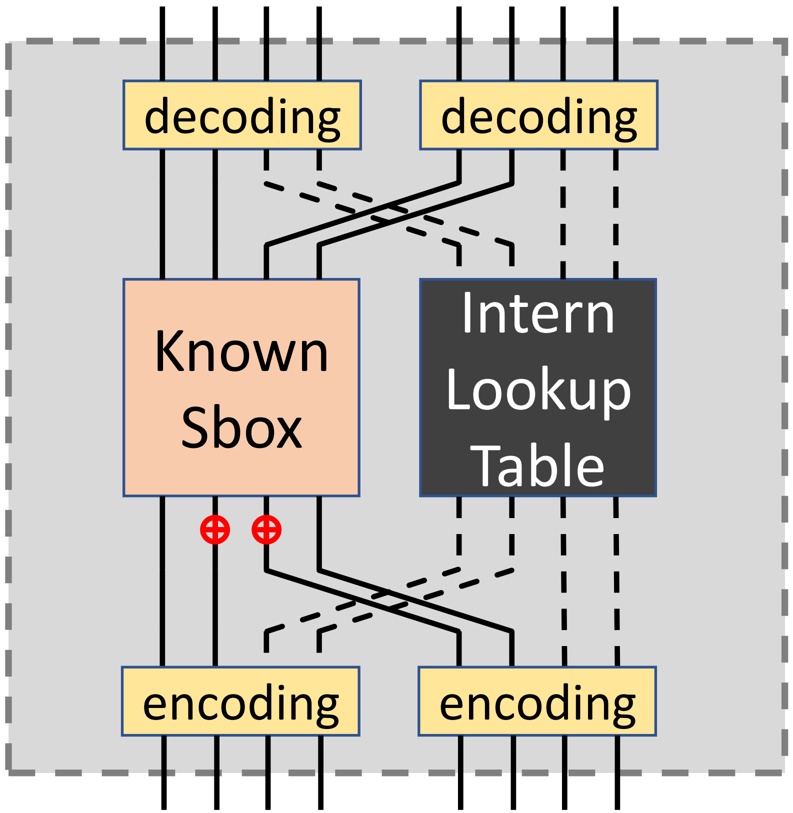
- Differential attack: If out_1 or out_2 have been modified, an attacker knows that only the first two output bits of the Sbox has been modified



Presentation of our encoding solution that avoids these limitations

Our solution resolves these weaknesses

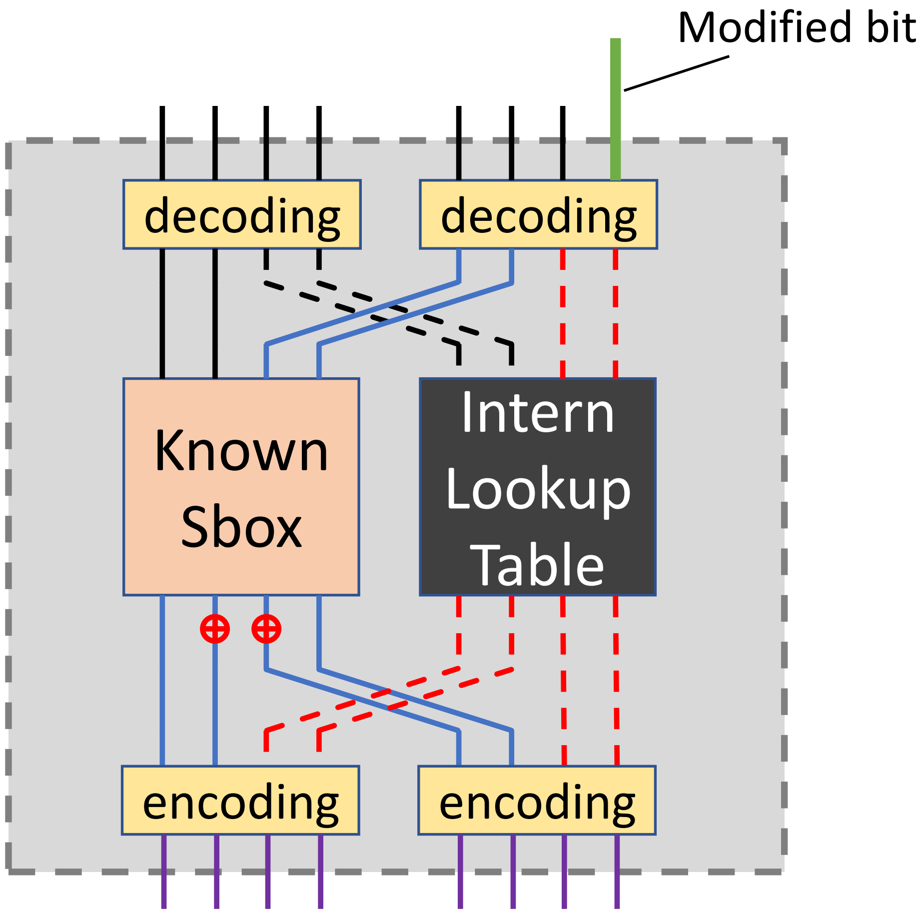
- Our solution involves random bits, that are represented in dashed lines
- These bits are used to encode the output
- They are updated with an arbitrary-chosen intern lookup table
- The resulting encoded table is called a Tbox



UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Our solution is resistant to brute-force and differential attacks

- In this example, there exists $\left((2^4)!\right)^5 \times 2^2 \approx 2^{223}$ possible Tboxes
- Modifying any input bit will have an overall impact on the output bits



Review of the encodability of the LWC finalists

The key must be spread throughout the algorithm

- The dispersion of the key throughout an algorithm forces a white-box attacker to study more parts of it
- The disclosure of the state allows an attacker to compute all following operations that are not key-dependant

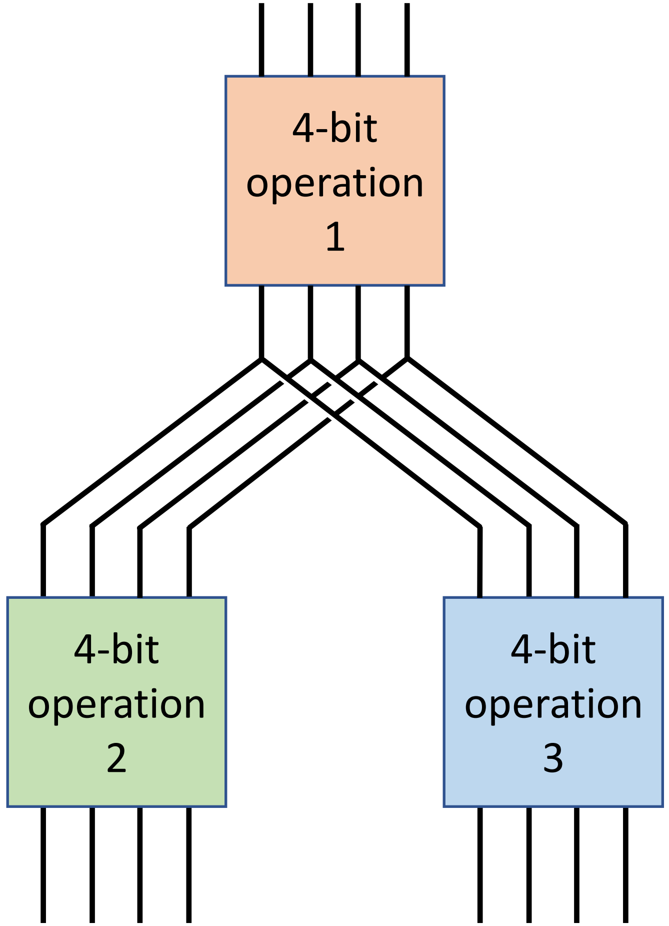
- For these reasons, we eliminated the following algorithms:
  - Isap
  - Ascon
  - Photon-Beetle
  - Sparkle
  - Xoodyak
  - Grain128-AEAD

UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Some algorithms are duplicating some state bits during computation

- If we encode an output being used more than once, it will imply that its corresponding decoding will be applied multiple times

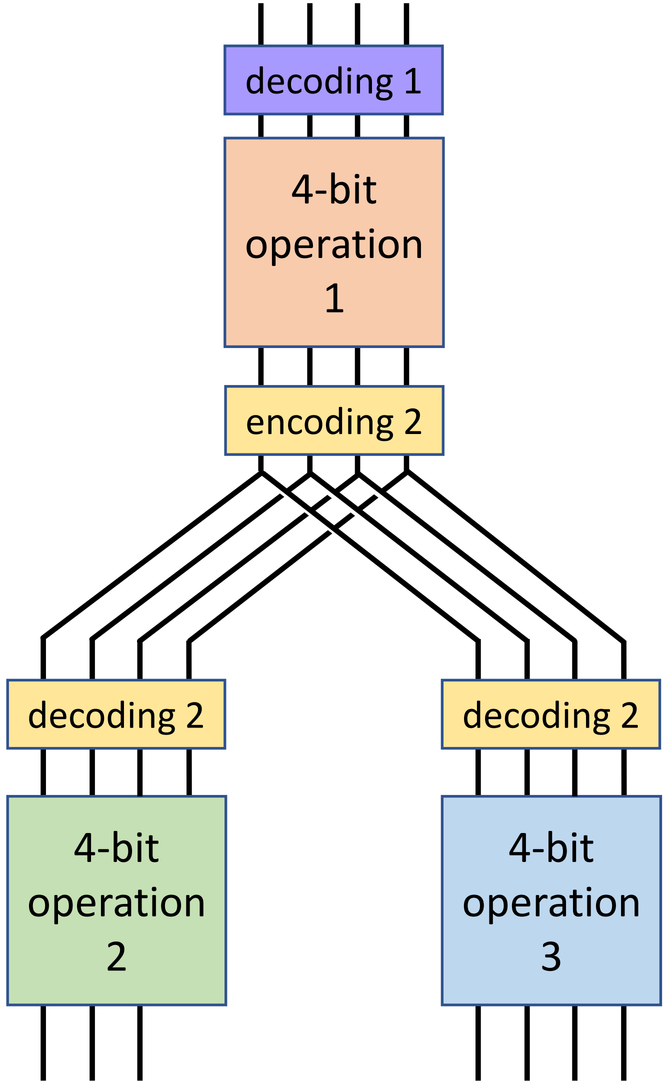
- This can give complementary information on this decoding



UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Some algorithms are duplicating some state bits during computation

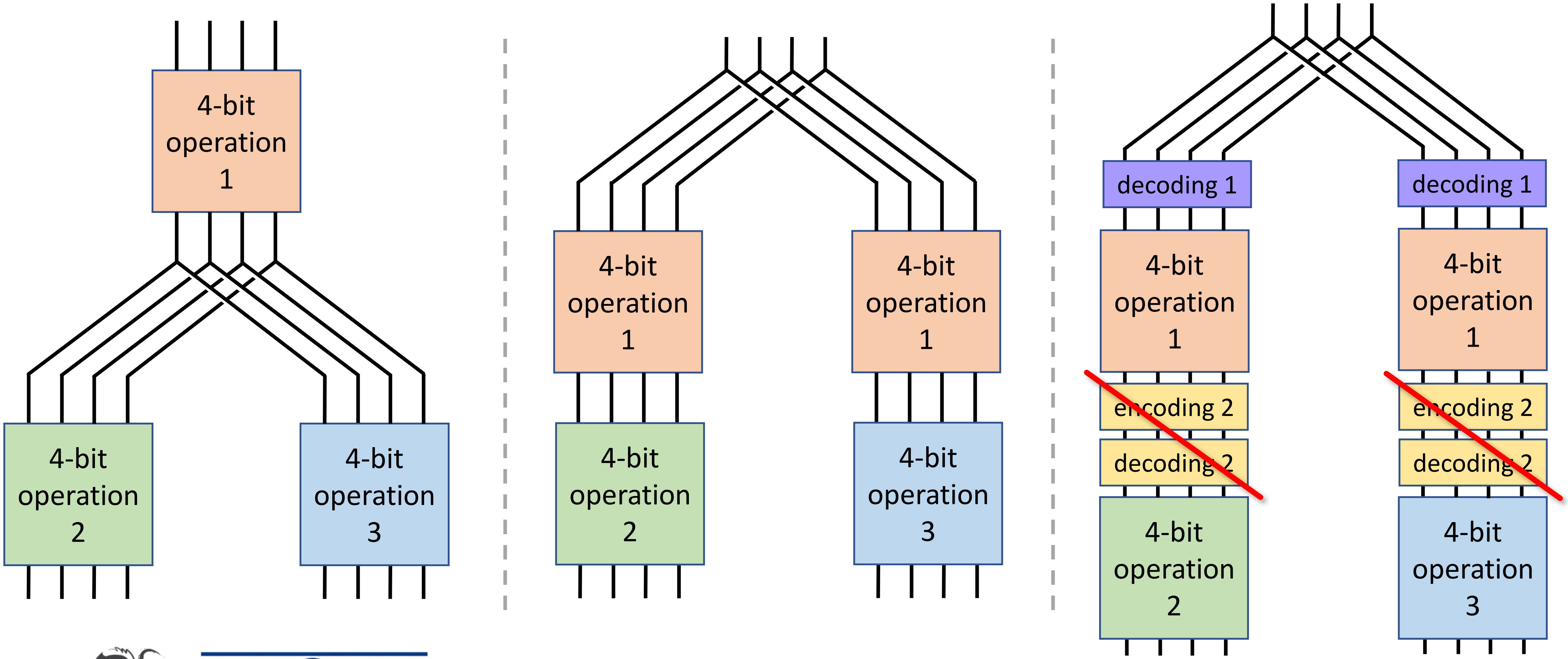
- If we encode an output being used more than once, it will imply that its corresponding decoding will be applied multiple times
- This can give complementary information on this decoding



UNIVERSITÉ DE RENNES 1

NAGRA KUDELSKI

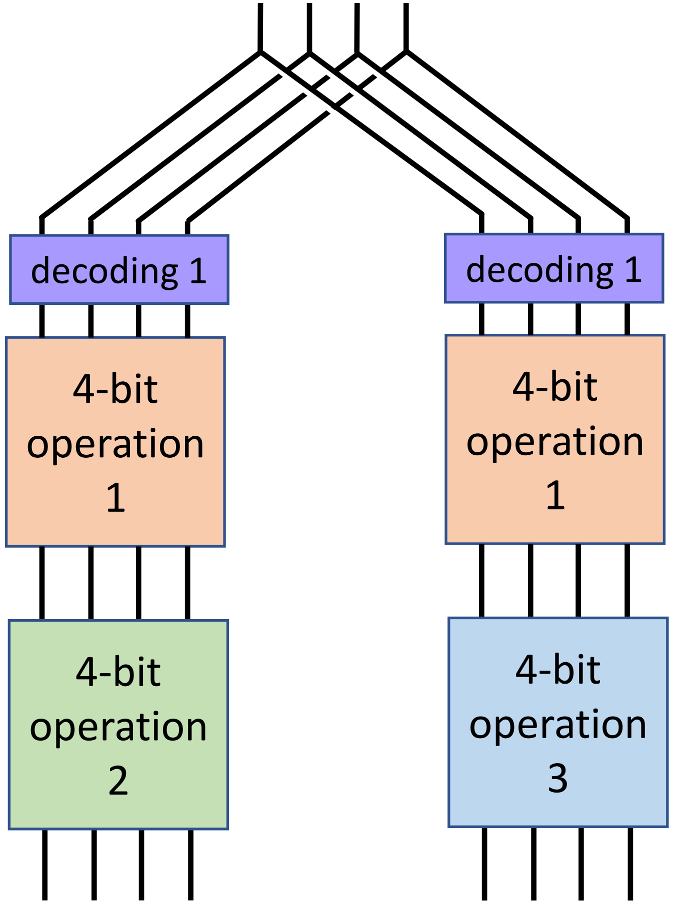
To avoid that, we can merge the operations



UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Merging operations is very heavy

- If a round operation needs to be merged, the state size will increase exponentially with the number of rounds.
- So, we want to avoid algorithms that are dependent on merging



UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

TinyJambu needs to merge operations

- The 128-bit state of TinyJambu is regarded as a 128-bit LFSR
- Each state bit can be used 5 times, so we need to merge the operations to avoid re-using the same encoding
- Because the LFSR is clocked up to 1024 times, merging operation would be too heavy

UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Romulus uses a too large XOR

- Romulus uses Skinny, that has 8-bit Sboxes, followed by 8-bit XORs.
- It would be too heavy to encode the XOR, as it has a 16-bit input

- Therefore, we need to split the output of the Sboxes onto two 4-bit groups, to have a following 8-bit input XOR
- To avoid encoding duplication, we need to merge round operations, which is too heavy

There are restrictions for an Elephant white-box implementation

- Elephant uses a function $mask_K^{a,b}$ which extends the key K, depending on block indexes of the message and associated data
- We want to precompute it to reduce the key manipulation

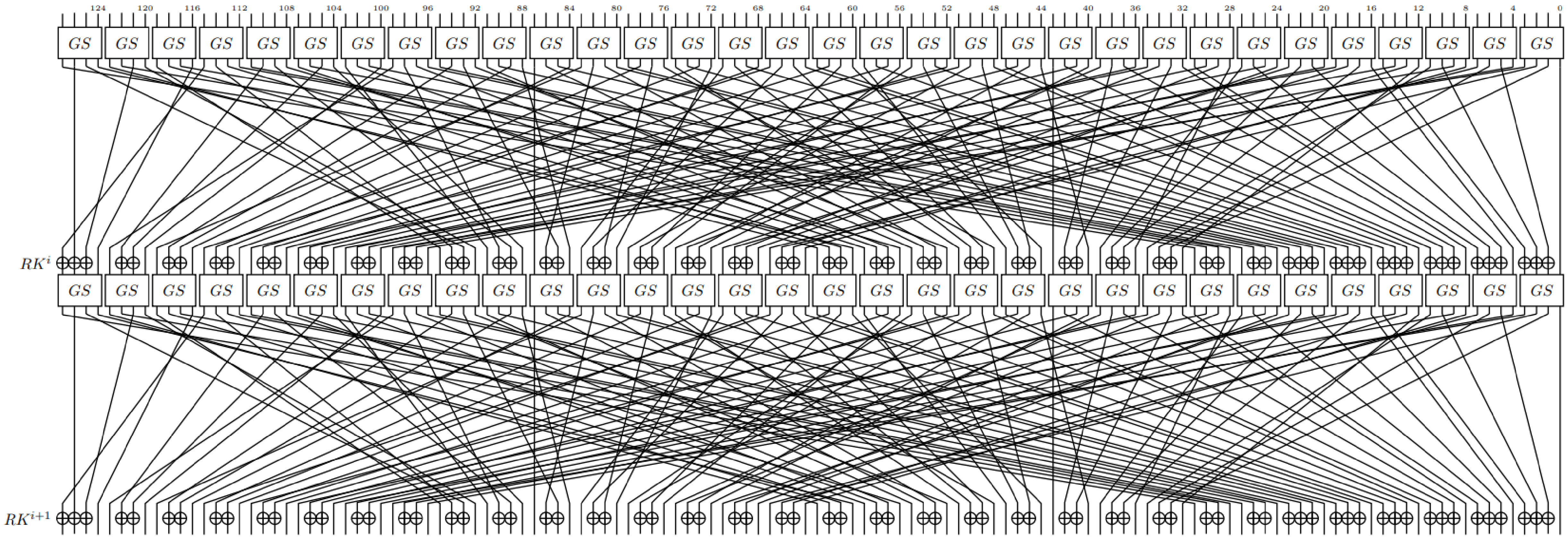
We must restrict message and associated data length in order to perform the precomputation

- However, if Elephant uses Spongent-π (and not Keccak), our solution can be applied in the same fashion as GIFT

UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Presentation of our solution applied to GIFT

Overview of GIFT-128

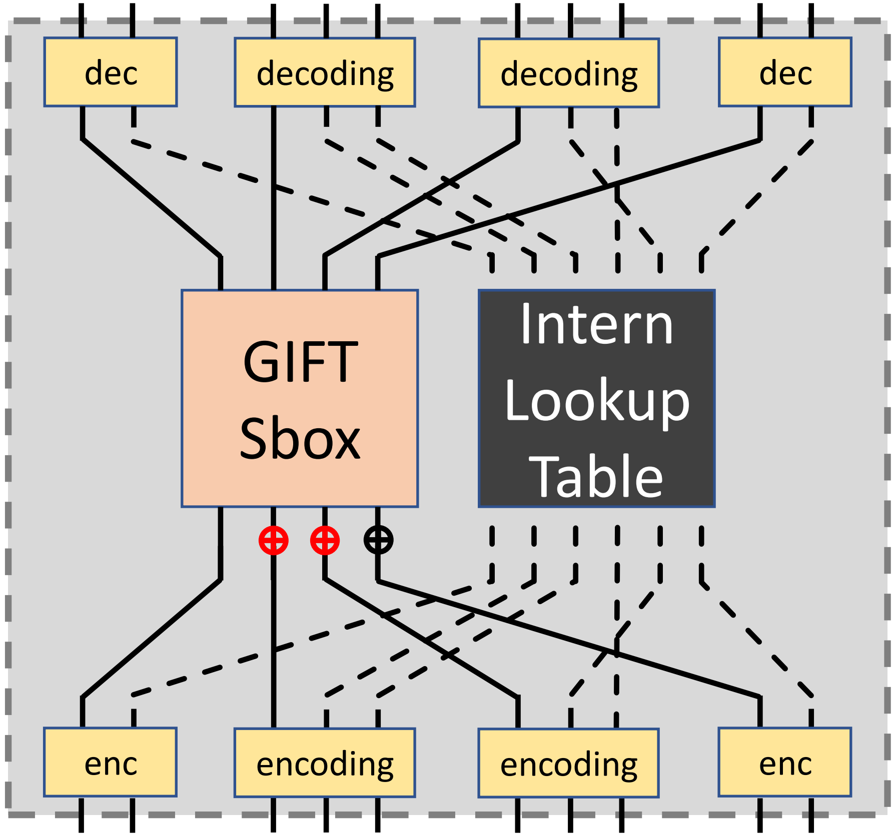


2 rounds of GIFT128, taken from GIFT-COFB specification

An encoded GIFT Sbox with our solution

- Has 375 bits of security
- Weighs 1.28 KB

- We can also use only one pseudo-random bit per encoding, for 80 bits of security, and a weight of 2048 bits



UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Comparison between our light white-box version of GIFT and a regular implementation

Nature of the tests	*GIFTEmbedded*	*GIFTEncoded*
Execution time (4000 runs)	15.34 ms	94.68 ms
Size of binary	132.2 kB	1.2 MB

On an 11th gen Intel Core i7-1185G7, using gcc

UNIVERSITÉ DE RENNES 1 NAGRA KUDELSKI

Thank you for our attention !

Questions ?

Alex Charles: alex.charles205@gmail.com
Chloé Gravouil: chloe.gravouil@nagra.com