Blind Side Channel on the Elephant LFSR

Awaleh Houssein Meraneh[1], Christophe Clavier[2], Hélène Le Bouder[1], Julien Maillard[2,3]
and Gaël Thomas[4]
[1]*IMT-Atlantique, OCIF, IRISA, Rennes, France*
[2]*Université de Limoges, XLIM-CNRS, Limoges, France*
[3]*Université de Grenoble Alpes, CEA, LETI MINATEC Campus, F-38054 Grenoble, France*
[4]*DGA Maîtrise de l'Information, Bruz, France*

Keywords: Blind Side Channel Analysis, Hamming Weight, Elephant, LFSR, NIST.

Abstract: Elephant is a finalist to the NIST lightweight cryptography competition. In this paper, the first theoretical blind side channel attack against the authenticated encryption algorithm Elephant is presented. More precisely, we are targetting the LFSR-based counter used internally. LFSRs are classic functions used in symmetric cryptography. In the case of Elephant, retrieving the initial state of the LFSR is equivalent to retrieving the encryption key. The paper ends by the study of different ways to tweak the design of Elephant to mitigate our attack.

1 INTRODUCTION

Internet of things (IoT) devices become more and more widespread within our day-to-day life. From military grade to general purpose hardware, the need for strong security raises. The cryptosystems implemented on those devices must ensure both security and low power consumption overhead. In this context, the *National Institute of Standards and Technology* (NIST) started the competition for lightweight cryptography candidates for authenticated encryption (NIST, 2018). An authenticated encryption algorithm should ensure confidentiality and integrity of the communications.

The security of authenticated encryption schemes can be supported by several strategies. Various approaches have been considered by the lightweight cryptography competition candidates:

- cryptographic permutation with sponge or duplex construction (Dobraunig et al., 2014; Beierle et al., 2019; Daemen et al., 2020),
- block cipher combined with a mode (e.g. AES combined with Galois/Counter Mode) (Iwata et al., 2020; Beyne et al., 2021),
- stream cipher paradigms (Hell et al., 2021).

When discussing about the security of a cryptographic algorithm, numerous tools allow the cryptographers to prove the security of a cipher. Unfortunately those tools do not consider the interaction of the computing unit with its physical environment. Physical attacks are a real threat, even for cryptographic algorithms proved secure mathematically. Physical attacks are divided in two families: Side-Channel Analysis (SCA) and the fault injection attacks.

Motivations

Many attacks exist on the different traditional cryptographic algorithms, for example on AES (Brier et al., 2004; Giraud, 2004). Lightweight cryptography, much younger and used in embedded devices, has been far less studied. For example, attacks on stream ciphers (Rechberger and Oswald, 2004) or sponge functions (Samwel and Daemen, 2017) are less common. That is why we chose to study SCA against new authenticated encryptions. The chosen algorithm is the cryptosystem Elephant (Beyne et al., 2021). More precisely, in this paper, the focus is about using *Linear Feedback Shift Registers* (LFSR), in a block cipher combined with a mode construction. Some attacks exist yet as in (Rechberger and Oswald, 2004; Joux and Delaunay, 2006; Burman et al., 2007; Chakraborty et al., 2014; Kazmi et al., 2017; Jurecek et al., 2019), but the main difference is in model of the attacker. To the best our knowledge, there is no blind side channel attack on LFSR in the context of authenticated encryption. So in this paper, a blind side channel attack targeting the LFSR of the Elephant algorithm, is presented.

Houssein Meraneh, A., Clavier, C., Le Bouder, H., Maillard, J. and Thomas, G.
Blind Side Channel on the Elephant LFSR.
DOI: 10.5220/0011135300003283
In *Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022)*, pages 25-34
ISBN: 978-989-758-590-6; ISSN: 2184-7711

Contribution

In this paper, we present the first theoretical blind side channel attack targeting the LFSR of the Elephant algorithm. We exploit the usage of intermediate variables that are statistically dependent to the secret (here the secret LFSR initial state) and show that this structure could eventually threaten the security of a cryptosystem's regarding side-channel analysis.

Also the study of the influence of the choice of the LFSR is presented.

Organisation

The paper is organized as follows. In section 2 the context about blind side channel attack and the Elephant are introduced. The theoretical attack is explained in section 3. Details of implementation attack are described in section 4. The section 5 presents experimental results and discussion about LFSR design. Finally, a conclusion is drawn in 6.

2 CONTEXT

In this section, first Elephant description is presented, then the context of blind SCA is introduced.

2.1 Elephant

An authenticated encryption algorithm should ensure confidentiality and integrity. It takes as input different parameters: a plaintext, data associated to the plaintext, a secret key, and an initialisation vector also called a nonce. The nonce is public but must be different for each new plaintext. The algorithm ensures confidentiality of the plaintext and integrity of both the plaintext and the associated data.

Elephant (Beyne et al., 2020; Beyne et al., 2021) is a nonce-based authenticated encryption with associated data (AEAD) finalists to the NIST lightweight cryptography competition. It is an Encrypt-then-MAC construction that combines CTR-mode encryption with a variant of the protected counter sum (Bernstein, 1999; Luykx et al., 2016).

It uses a cryptographic permutation masked with LFSRs in an Even-Mansour-like fashion (Granger et al., 2016) in place of a blockcipher.

Let P be an n-bit cryptographic permutation, and φ an n-bit LFSR. Let the function $\mathsf{mask} : \{0,1\}^{128} \times \mathbb{N} \times \{0,1,2\} \to \{0,1\}^n$ be defined as follows:

$$\mathsf{mask}_K^{i,b} = (\varphi \oplus \mathsf{id})^b \circ \varphi^i \circ \mathsf{P}(K||0^{n-128}) \qquad (4)$$

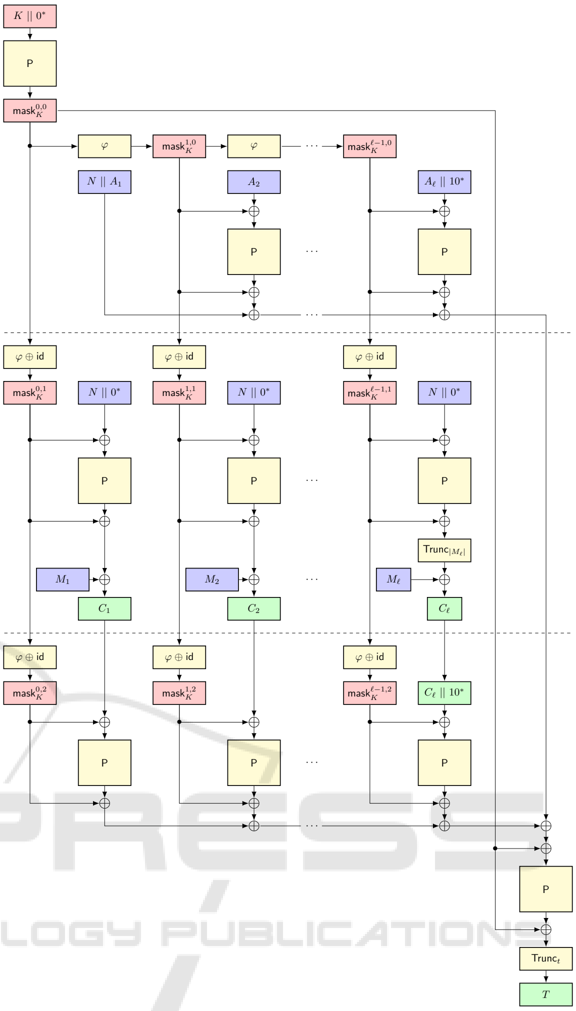


Figure 1: Elephant associated data authentication (top), plaintext encryption (middle), and ciphertext authentication (bottom).

Let $\mathsf{Split}(X)$ be the function that splits the input X into n-bit blocks, where the last block is zero-padded. Let $\mathsf{Trunc}_t(X)$ be the t left-most bits of X.

Encryption `enc` under Elephant gets as input a 128-bit key K, a 96-bit nonce N, associated data $A \in \{0,1\}^*$, and a plaintext $M \in \{0,1\}^*$. It outputs a ciphertext C as large as M, and a t-bit tag T. The description of `enc` is given in Algorithm 1 and is depicted on Figure 1.

Decryption `dec` gets as input a 128-bit key K, a 96-bit nonce N, associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$, and t-bit tag T. It outputs a plaintext M as large as C if the tag T is correct, or the symbol $\perp$ otherwise. The description of `dec` easily follows from that of `enc`.

Elephant comes in three flavours which differ on the n-bit cryptographic permutation P and the LFSR φ used, as well as the tag size t.

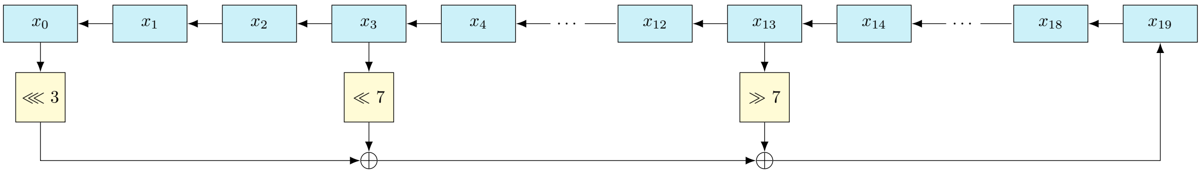


Figure 2: 160-bit LFSR φ_{Dumbo}.

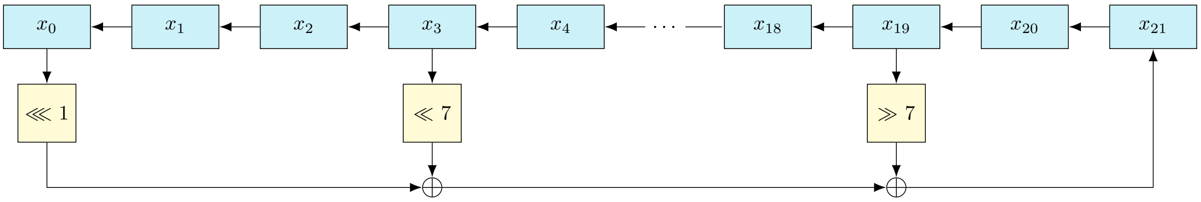


Figure 3: 176-bit LFSR φ_{Jumbo}.

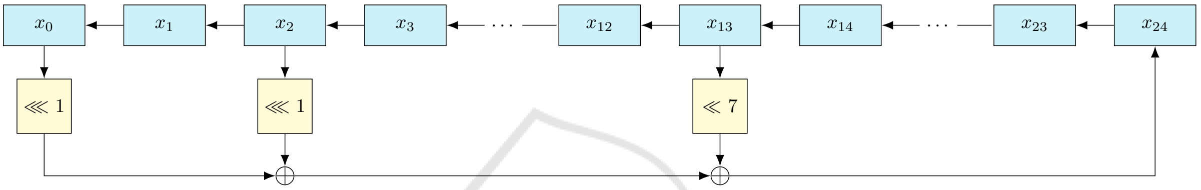


Figure 4: 200-bit LFSR $\varphi_{\mathsf{Delirium}}$.

$$\varphi_{\mathsf{Dumbo}} : (x_0, \cdots, x_{19}) \mapsto (x_1, \cdots, x_{19}, x_0 \lll 3 \oplus x_3 \ll 7 \oplus x_{13} \gg 7) \tag{1}$$

$$\varphi_{\mathsf{Jumbo}} : (x_0, \cdots, x_{21}) \mapsto (x_1, \cdots, x_{21}, x_0 \lll 1 \oplus x_3 \ll 7 \oplus x_{19} \gg 7) \tag{2}$$

$$\varphi_{\mathsf{Delirium}} : (x_0, \cdots, x_{24}) \mapsto (x_1, \cdots, x_{24}, x_0 \lll 1 \oplus x_2 \lll 1 \oplus x_{13} \ll 7) \tag{3}$$

Algorithm 1: Elephant encryption algorithm enc.

```
Input: $(K,N,A,M) \in \{0,1\}^{128} \times \{0,1\}^{96} \times \{0,1\}^* \times \{0,1\}^*$
Output: $(C,T) \in \{0,1\}^{|M|} \times \{0,1\}^t$
$M_1, \cdots, M_{\ell_M} \leftarrow \mathsf{Split}(M)$
for $i \leftarrow 1$ to $\ell_M$ do
    $C_i \leftarrow M_i \oplus \mathsf{P}(N||0^{n-96} \oplus \mathsf{mask}_K^{i-1,1}) \oplus \mathsf{mask}_K^{i-1,1}$
end for
$C \leftarrow \mathsf{Trunc}_{|M|}(C_1||\cdots||C_{\ell_M})$
$T \leftarrow 0^n$
$A_1, \cdots, A_{\ell_A} \leftarrow \mathsf{Split}(N||A||1)$
$C_1, \cdots, C_{\ell_C} \leftarrow \mathsf{Split}(C||1)$
$T \leftarrow A_1$
for $i \leftarrow 2$ to $\ell_A$ do
    $T \leftarrow T \oplus \mathsf{P}(A_i \oplus \mathsf{mask}_K^{i-1,0}) \oplus \mathsf{mask}_K^{i-1,0}$
end for
for $i \leftarrow 1$ to $\ell_C$ do
    $T \leftarrow T \oplus \mathsf{P}(C_i \oplus \mathsf{mask}_K^{i-1,2}) \oplus \mathsf{mask}_K^{i-1,2}$
end for
$T \leftarrow \mathsf{P}(T \oplus \mathsf{mask}_K^{0,0}) \oplus \mathsf{mask}_K^{0,0}$
return $(C, \mathsf{Trunc}_t(T))$
```

Dumbo uses the 160-bit permutation Spongent-π[160] (Bogdanov et al., 2011), the LFSR φ_{Dumbo} given by equation 1 and illustrated on Figure 2, and has tag size $t = 64$ bits.

Jumbo uses the 176-bit permutation Spongent-π[176] (Bogdanov et al., 2011), the LFSR φ_{Jumbo} given by equation 2 and illustrated on Figure 3, and has tag size $t = 64$ bits.

Delirium uses the 200-bit permutation Keccak-f[200] (Bertoni et al., 2011; NIST, 2015), the LFSR $\varphi_{\mathsf{Delirium}}$ given by equation 3 and illustrated on Figure 4, and has tag size $t = 128$ bits.

The three n-bit LFSRs used for the variants of Elephant are the GF(2)-linear maps given at the byte-level by the equations (1), (2) and (3).

2.2 Blind Side Channel Analysis

Even if an algorithm has been proven to be mathematically secure, its implementation can open the gate to the so-called physical attacks. (SCA) are a subcategory of physical attacks. They exploit the fact that some physical values of a device depend on intermediate values of the computation.

This is the so-called leakage of information of the circuit. It could be used to retrieve secrets, as a secret key.

Different kind of leakage can be exploited as times (Handschuh and Heys, 1998), power consumption or electromagnetic radiations (Standaert, 2010). In this paper, the leakage is power consumption. At each instant, the measurement of the intensity of the electric current reflects the activity of the circuit. The power consumption of a device is the sum of the power consumptions of each of its logic gates. An attacker therefore has the possibility of distinguishing a transition from 0 to 1 from a transition from 1 to 0.

A SCA is often leaded with a divide-and-conquer approach. Namely, the secret is divided into small pieces that are analysed independently. Different kind of analysis exist.

The *Simple Power Analysis* (SPA) (Mangard, 2002) are called simple because they determine directly, from an observation of the power consumption, during a normal execution of an algorithm, information on the calculation performed or the data manipulated.

The family of *Correlation Power Analysis* (CPA) (Kocher et al., 1999; Brier et al., 2004; Gierlichs et al., 2008) uses a mathematical model for the leakage. A confrontation between measurement and model is performed. More precisely, a statistic tool called distinguisher gives score to the different targets.

Template attacks are statistical categorizations (Chari et al., 2002). No mathematical model is required.

The blind side channel analysis family is new improvement in SCA (Joux and Delaunay, 2006; Burman et al., 2007; Chakraborty et al., 2014; Linge et al., 2014; Le Bouder et al., 2016; Clavier and Reynaud, 2017). The main idea is to not use the data as plaintext or ciphertext. Only the leakage is used. The power consumption leakage is very correlated to the *Hamming Weight* (HW) of the data. So in blind SCA, a strong assumption is make. The attacker can retrieve HW with the leakage. More precisely, the power consumption can be seen as a noisy HW. In this paper, the considered adversary model is that: the HW of bytes can be obtained by an attacker. This model is made feasible by the fact that the attacker can average power traces.

The first theoretical result on SCA on LFSRs are (Joux and Delaunay, 2006) where the authors leverage the dependence between the leakage and the unique feedback bit of a Galois LFSR. The case of Fibonacci LFSRs where a single new value is computed at each iteration is studied in (Burman et al., 2007). Finally, both kind of LFSRs are theoretically and practically compared in (Chakraborty et al., 2014).

3 THEORETICAL ATTACK

3.1 Goal

LFSRs are used in different lightweight cryptography candidates, and its initial state often depends on both the key and the nonce. As the nonce needs to be changed for each encryption, attacks on such schemes are limited to the decryption algorithm. In the case of Elephant, the LFSR only depends on the secret key, Consequently, our attack can be applied in an encryption scenario.

The goal of the presented attack is to retrieve the LFSR secret initial state. One has to remark three important points.

- Retrieving the initial state of the LFSR which is equal to $\mathsf{mask}_K^{0,0}$ is equivalent to retrieving the secret key. Indeed, the initial state is just the result of the known permutation P applied to the key.
- As the retroaction polynomial is publicly known, it is possible to shift the LFSR backwards: an attacker who knows 20 consecutive bytes of the secret stream is able to recover the initial state.
- The smaller the LFSR is, the more the attack is able to succeed. As a consequence, the Dumbo 2 instance is the most vulnerable instance: the focus is on Dumbo in the following of this paper.

3.2 Leakage in the LFSR

In this attack, it is assumed that the Hamming Weight of all bytes of the LFSR can be obtained by an attacker.

Let x be a byte, so x can take 256 values in $[\![0,255]\!]$. With the HW of x, the attacker reduces the list of possible values, as shown in Table 1.

Table 1: Number of possible values for an Hamming weight.

$HW(x)$	0	1	2	3	4	5	6	7	8
$\#x$	1	8	28	56	70	56	28	8	1

Since the LFSR generates a single new byte at each iteration, let $x_0, \cdots, x_{19}$ be the content of the Dumbo LFSR initialised with $\mathsf{mask}_K^{0,0}$, and extend the notation for $j \geq 20$, by letting x_j be the new byte generated at iteration $j-20$. In other words, the attacker has the following relation (L1).

$$x_{j+20} = (x_j \lll 3) \oplus (x_{j+3} \ll 7) \oplus (x_{j+13} \gg 7). \tag{L1}$$

The first idea is to use the knowledge of the following Hamming weights: $HW\left(x_{j+20}\right)$, $HW\left(x_j\right) = HW(x_j \lll 3)$, $HW\left(x_{j+3}\right)$ and $HW\left(x_{j+13}\right)$.

Moreover, one has to remark that:

$$HW\left(x_j\right) = HW\left(x_j \lll 3\right). \tag{5}$$

So with the two equations (L1) and (5) the attacker has:

$$HW\left(x_{j+20}\right) = \begin{cases} HW\left(x_j\right) \\ HW\left(x_j\right) + 1 \\ HW\left(x_j\right) - 1 \\ HW\left(x_j\right) + 2 \\ HW\left(x_j\right) - 2 \end{cases} \tag{6}$$

Looking more precisely at equation (L1), it can be seen that the difference $HW(x_{j+20}) - HW(x_j)$ depends on only four bits. Letting $x_j[i]$ denote the i-th least significant bit of byte x_j, these four bits are $\{x_{j+3}[0]; x_{j+13}[7]; x_j[4]; x_j[5]\}$. Table 2 gives the value of observed difference $HW(x_{j+20}) - HW(x_j)$ depending on the values of these four bits. In the worst case, there are only 6 possibilities left, out of 16.

Table 2: Values of $HW\left(x_{j+20}\right) - HW\left(x_j\right)$ according to $\{x_{j+3}[0]; x_{j+13}[7]; x_j[4]; x_j[5]\}$.

$HW\left(x_{j+20}\right) - HW(x_j)$		$(x_{j+3}[0], x_{j+13}[7]) =$			
		$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
$(x_j[4], x_j[5]) =$	$(0,0)$	0	+1	+1	+2
	$(1,0)$	0	+1	−1	0
	$(0,1)$	0	−1	+1	0
	$(1,1)$	0	−1	−1	−2

3.3 Link between the Different Masks

The value $\mathsf{mask}_K^{i,1}$ can be expressed in terms of $\mathsf{mask}_K^{*,0}$ as in (7).

$$\begin{aligned} \mathsf{mask}_K^{i,1} &= (\varphi \oplus \mathsf{id})\left(\mathsf{mask}_K^{i,0}\right) \\ &= \varphi\left(\mathsf{mask}_K^{i,0}\right) \oplus \mathsf{mask}_K^{i,0} \\ &= \mathsf{mask}_K^{i+1,0} \oplus \mathsf{mask}_K^{i,0}. \end{aligned} \tag{7}$$

Likewise, for $\mathsf{mask}_K^{i,2}$, equation (8) holds.

$$\begin{aligned} \mathsf{mask}_K^{i,2} &= (\varphi \oplus \mathsf{id})^2\left(\mathsf{mask}_K^{i,0}\right) \\ &= (\varphi^2 \oplus \mathsf{id})\left(\mathsf{mask}_K^{i,0}\right) \\ &= \varphi^2\left(\mathsf{mask}_K^{i,0}\right) \oplus \mathsf{mask}_K^{i,0} \\ &= \mathsf{mask}_K^{i+2,0} \oplus \mathsf{mask}_K^{i,0} \end{aligned} \tag{8}$$

As in the case of $\mathsf{mask}_K^{i,0}$, let y_j denote either the byte j of $\mathsf{mask}_K^{0,1}$ when $0 \leq j \leq 19$, or the new byte obtained after $j-20$ iterations of the LFSR initialised with $\mathsf{mask}_K^{0,1}$. Likewise, let z_j denote either the byte j of $\mathsf{mask}_K^{0,2}$ when $0 \leq j \leq 19$, or the new byte obtained after $j-20$ iterations of the LFSR initialised with $\mathsf{mask}_K^{0,2}$.

Equation (7) then translates to equation (9).

$$y_j = x_j \oplus x_{j+1} \tag{9}$$

Likewise, the equation 8 translates to (10).

$$z_j = x_j \oplus x_{j+2}. \tag{10}$$

The evolutions of the LFSR are analogous to (L1):

$$y_{j+20} = (y_j \lll 3) \oplus (y_{j+3} \ll 7) \oplus (y_{j+13} \gg 7). \tag{11}$$

$$z_{j+20} = (z_j \lll 3) \oplus (z_{j+3} \ll 7) \oplus (z_{j+13} \gg 7). \tag{12}$$

The attacker can thus exploit two attack vectors: on the one hand, equations (L1), (11), and (12) coming from iterating the LFSR, and on the other hand, equations (9) and (10) coming from the different masks used for domain separation.

4 ATTACK STRATEGY

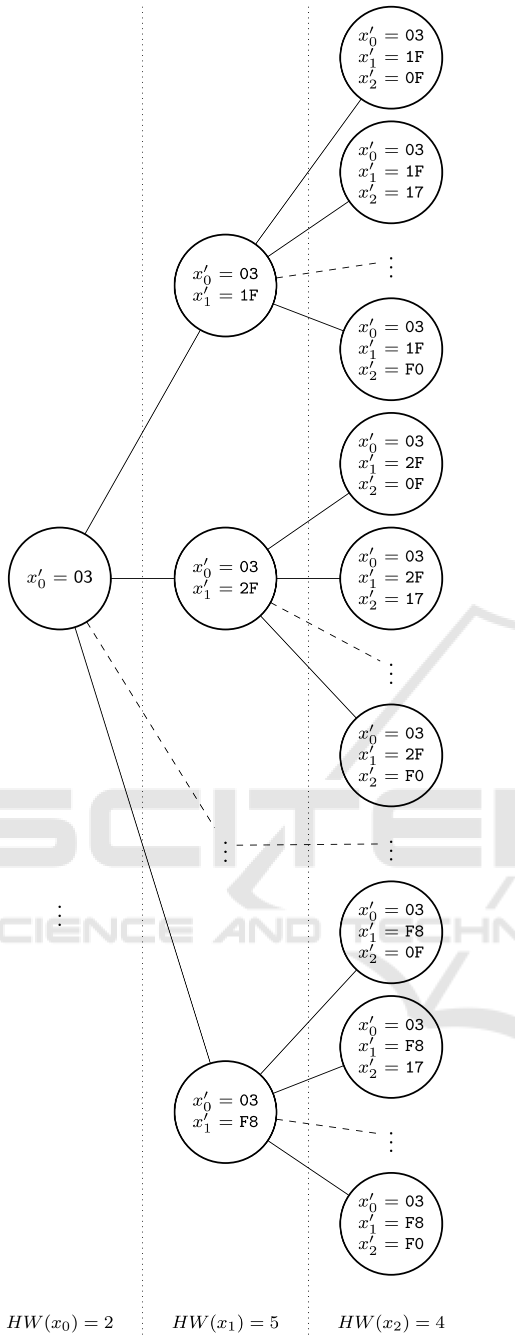
The whole search space corresponding to the initial state of the LFSR is represented as a rooted tree. The nodes at depth j correspond to the all possible values for the bytes x_0 to x_j of the LFSR. The tested candidates are denoted by $(x'_0, \cdots, x'_{19})$. The nodes in the graph of the search space are labelled as follows:

- the nodes at depth j correspond to the all possible values of $(x'_0, \cdots, x'_j)$;
- the children of node $(x'_0, \cdots, x'_j)$, are the nodes labelled: $(x'_0, \cdots, x'_j, x'_{j+1})$ for all values of x'_{j+1}.

In practice, to reduce the number of nodes, only the nodes having the correct Hamming weights are considered. In other words, it suffices to consider nodes with $HW(x'_j) = HW(x_j)$. An example of such tree is given on Figure 5.

A backtracking algorithm is used. The tree is traversed in a depth-first manner. For each step, the attacker tests whether the current candidate $(x'_0, \cdots, x'_j)$ satisfies the different conditions given by the observed Hamming weights. This test is given by algorithm 2.

If the test succeeds, the algorithm goes down to the next layer to test the values of the byte x'_{j+1}. If it reaches the bottom of the tree, then a good candidate



$x'_0 = 03$
$x'_0 = 03$ $x'_1 = 1F$
$x'_0 = 03$ $x'_1 = 1F$ $x'_2 = 0F$
$x'_0 = 03$ $x'_1 = 1F$ $x'_2 = 17$
$x'_0 = 03$ $x'_1 = 1F$ $x'_2 = F0$
$x'_0 = 03$ $x'_1 = 2F$
$x'_0 = 03$ $x'_1 = 2F$ $x'_2 = 0F$
$x'_0 = 03$ $x'_1 = 2F$ $x'_2 = 17$
$x'_0 = 03$ $x'_1 = 2F$ $x'_2 = F0$
$x'_0 = 03$ $x'_1 = F8$
$x'_0 = 03$ $x'_1 = F8$ $x'_2 = 0F$
$x'_0 = 03$ $x'_1 = F8$ $x'_2 = 17$
$x'_0 = 03$ $x'_1 = F8$ $x'_2 = F0$
$HW(x_0) = 2$
$HW(x_1) = 5$
$HW(x_2) = 4$

Figure 5: Example of the tree representation of the LFSR initial state for the Hamming weights given at the bottom. Only the first three layers of the subtree rooted at $x'_0 = 03$ are shown.

has been found, and can be saved. The algorithm then iterates upon the next untested node.

If, at some point, the Hamming weights conditions do not hold for the current (partial) candidate, then no node in the sub-tree rooted at that node can lead to a good candidate. Thus, it can be pruned from the whole tree, saving the cost of searching it. Finally, the algorithm ends when the whole tree has been searched. A pseudocode of the attack is given by algorithm 3.

Algorithm 2: $\mathsf{isvalid}(X'_0, \cdots, x'_j)$.

```
Input: Byte-wise partial candidate $(x'_0, \cdots, x'_j)$ of
    length $1 \leq j+1 \leq 20$
    Assumes $\mathsf{isvalid}(x'_0, \cdots, x'_{j-1})$ is true.
Output: true if candidate $(x'_0, \cdots, x'_j)$ is compatible
    with the observations, false otherwise

# Hamming weights of the xors
if $HW(x'_j) \neq HW(x_j)$ then
    return false
end if
if $HW(x'_j \oplus x'_{j-1}) \neq HW(y_{j-1})$ then
    return false
end if
if $HW(x'_j \oplus x'_{j-2}) \neq HW(z_{j-2})$ then
    return false
end if

# Hamming weights of the feedbacks
if $|HW(x'_j \lll 3) - HW(x_{j+20})| > 2$ then
    return false
end if
if $|HW(x'_{j-3} \lll 3 \oplus x'_j \ll 7) - HW(x_{j+17})| > 1$
then
    return false
end if
if $HW(x'_{j-13} \lll 3 \oplus x'_{j-10} \ll 7 \oplus x'_j \gg 7) \neq$
$HW(x_{j+7})$ then
    return false
end if

return true
```

To improve the efficiency of algorithm 3, the attacker can overlook some of the first iterations of the LFSR, and start the attack at a time they deem more satisfying. Indeed, the complexity of the attack depend on the number of nodes visited in the tree. Looking only at the first four layers, that number, denoted by N_4, can be expressed using binomial coefficients as a function of $(HW(x_0), HW(x_1), HW(x_2), HW(x_3))$:

$$N_4 = \binom{8}{HW(x_0)}\binom{8}{HW(x_1)}\binom{8}{HW(x_2)}\binom{8}{HW(x_3)}.$$

The idea is then to look at the quadruplet at the next iteration, namely $(HW(x_1), HW(x_2), HW(x_3), HW(x_4))$, and so on, until the number of nodes in the first four layers of the tree is sufficiently small. One has to remark that the number 4 is arbitrary here, and the attacker

Algorithm 3: Attack.

Input: Observed Hamming weights $HW(x_0), \cdots, HW(x_{19})$, $HW(y_0), \cdots, HW(y_{18})$, and $HW(z_0), \cdots, HW(z_{17})$. For the sake of clarity, they are seen as global variables.

Output: S set of keys compatible with the observed Hamming weights

```
$(x'_0, \cdots, x'_{19}) \leftarrow (0, \cdots, 0)$
$\ell \leftarrow 0$
$S \leftarrow \{\}$
while true do
    if $j < 19$ and isvalid$(x'_0, \cdots, x'_j)$ then
        $j \leftarrow j + 1$
        $x'_j \leftarrow 0$
    else
        if $j = 19$ and isvalid$(x'_0, \cdots, x'_j)$ then
            $S \leftarrow S \cup \{(x'_0, \cdots, x'_j)\}$
        end if
        while $j \geq 0$ and $x'_j = \mathtt{FF}$ do
            $j \leftarrow j - 1$
        end while
        if $j \geq 0$ then
            $x'_j \leftarrow x'_j + 1$
        else
            break
        end if
    end if
end while
return $S$
```

can choose whatever value they might prefer.

The question is now, what bound on N_4 does the attacker choose? We have chosen to fix a threshold at $N_4 \leq 1,756,160$. With this, about 25% of the all quadruplets are kept. Luckily, the probability to find such a quadruplet in the LFSR rapidly increases to one when iterating, because of the good statistical properties of LFSRs.

5 RESULTS AND DISCUSSION

5.1 Elephant Attack

We have simulated the attack on $N_{runs} = 560$ randomly generated Dumbo keys. For each, the number N_{nodes} of nodes effectively traversed in the tree has been counted. This number roughly corresponds to the time complexity of the attack. Among those nodes, we have specifically counted the number N_{keys} of nodes on the last layer; i.e. nodes that correspond to plausible guesses that remain to be brute forced to finish the attack.

Unfortunately, only just above half (53.57%) of the runs have ended after two days, and about a quarter has been still running after a week. On average, for the runs that finished after two days, the number of nodes traversed is $N_{nodes} = 2^{41.82}$, and the number of remaining keys is $N_{keys} = 2^{36.59}$.

5.2 Impact of the Generation of Masks

The natural question is about what could be done to mitigate this attack. Outside of using generic countermeasures, like e.g. boolean masking, there seem to be two possibilities for improvement. Indeed, the attacker gains information from two sources:

- from equations (7) and (8) used to derive the masks for domain separation;
- from the LFSR state update equation (L1).

Thus either the mask derivation, or the LFSR can be changed, or both. This section studies the former case.

We ran two experiments, similar to that in section 5.1 except that the attacker does not gain information on every Hamming weights. In the first experiment, they only know the values of the $HW(x_j)$, and the $HW(y_j)$ for sufficiently j. In other words, compared to the experiment in section 5.1, they lost the knowledge of the $HW(z_j)$. Likewise, in the second experiment, they only know the values of the $HW(x_j)$ for sufficiently j. Unfortunately, in both cases, none of the $N_{runs} = 120$ runs done has terminated after a week.

From these experiments, it seems that the knowledge of the $HW(x_j)$, $HW(y_j)$, and $HW(z_j)$ contributed heavily on the success of the attack. It would then seem a good idea to tweak the cryptographic mode of operation by finding another way of generating masks for domain separation.

5.3 Studies on Different LFSRs

This section is dedicated to the study of the influence of the choice of the LFSR. To keep the spirit of the original Elephant, only Fibonacci-like LFSRs, at the byte level, are considered. More specifically, LFSRs considered are: LFSRs where a single new byte is computed from a combination of three bytes using byte-wise shifts and rotations. As usual, the associated feedback polynomial must be primitive to ensure only maximum-length sequences can be generated.

Among all possible candidates, different behaviours can be triggered.

In this paper, the **type** of an LFSR is defined as the sequence of number of bits unknown to the attaquer at each depth in the tree where a new feedback occurs.

Looking at equation (L1), it can be seen that:

$$\left|HW(x_{j+20}) - HW(x_j \lll 3)\right| \leq 2$$

since there are only 2 bits that are modified by:

$$x_{j+3} \ll 7 \oplus x_{j+13} \gg 7.$$

Thus, if other feedback function are used, with more bits involved, it can be expected to have an impact on the attack.

Later in the attack, when at depth 3 in the tree, the same idea can be applied to check whether:

$$\left|HW(x_{j+17}) - HW(x_{j-3} \lll 3 \oplus x_j \ll 7)\right| \leq 1$$

since now only the single bit $x_{j+10} \gg 7$ is unknown. In conclusion, the type of the Dumbo LFSR is $[2,1]$.

LFSR with different types can be a first criterion when testing our attack.

A second criterion can be: how far apart the feedback bytes are. Indeed, the tighter they are, the faster the attacker can use equation (L1) at its full potential. In the case of Dumbo, the feedback bytes are at indices 0, 3, and 13. We call 13 the **depth**, this is simply the highest index of the feedbacks.

We chose LFSR based on these two criteria. Types is defined from $[2,1]$ to $[8,8]$. For types $[2,1]$, and $[5,*]$, we looked at all the possible LFSRs in order to study the influence of their depth.

The state update function of the different LFSR tested are given by equations (L2) to (L21). Their type and depth are given at the second, respectively third, column of Table 3.

L2 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+1} \ll 7 \oplus x_{j+11} \gg 7$

L3 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+14} \gg 3 \oplus x_{j+17} \gg 7$

L4 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+3} \gg 3 \oplus x_{j+13} \gg 7$

L5 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+9} \gg 3 \oplus x_{j+15} \gg 7$

L6 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+9} \ll 4 \oplus x_{j+19} \gg 7$

L7 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+1} \ll 5 \oplus x_{j+3} \gg 6$

L8 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+4} \gg 3 \oplus x_{j+19} \gg 5$

L9 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+7} \gg 3 \oplus x_{j+18} \gg 5$

L10 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+3} \gg 3 \oplus x_{j+9} \gg 5$

L11 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+1} \gg 7 \oplus x_{j+17} \ll 4$

L12 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+5} \gg 7 \oplus x_{j+19} \gg 3$

L13 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+5} \ll 7 \oplus x_{j+16} \ll 3$

L14 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+1} \gg 7 \oplus x_{j+9} \gg 3$

L15 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+13} \ll 5 \oplus x_{j+19} \ll 3$

L16 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+14} \gg 7 \oplus x_{j+17} \gg 3$

L17 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+4} \ll 1 \oplus x_{j+5} \gg 6$

L18 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+3} \gg 1 \oplus x_{j+9} \ll 1$

L19 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+4} \gg 1 \oplus x_{j+5} \ll 1$

L20 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+1} \gg 1 \oplus x_{j+8} \lll 7$

L21 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+3} \ll 5 \oplus x_{j+4} \lll 5$

We ran the same experiment as in section 5.1 for every considered LFSR with $N_{tests} = 120$. For each LFSR, we noted the proportion of runs finished after two days of computations, the average number of nodes effectively traversed in the tree, and average number of remaining keys. Results are summarized in Table 3.

Table 3: Type, depth, proportion of runs finished after two days of computations, average number of nodes traversed, and number of remaining keys for Dumbo (L1), and LFSRs (L2) to (L21).

LFSR	type	depth	finished	N_{nodes}	N_{keys}
(L1)	$[2,1]$	13	53.57%	$2^{41.82}$	$2^{36.59}$
(L2)	$[2,1]$	11	82.5%	$2^{41.23}$	$2^{36.39}$
(L3)	$[5,1]$	17	0.83%	$2^{42.89}$	$2^{34.68}$
(L4)	$[5,1]$	13	94.17%	$2^{39.68}$	$2^{33.68}$
(L5)	$[5,1]$	15	28.33%	$2^{42.13}$	$2^{35.25}$
(L6)	$[5,1]$	19	11.67%	$2^{42.38}$	$2^{36.77}$
(L7)	$[5,2]$	3	100.0%	$2^{30.93}$	$2^{24.93}$
(L8)	$[5,3]$	19	0.83%	$2^{43.99}$	$2^{37.59}$
(L9)	$[5,3]$	18	0.0%	–	–
(L10)	$[5,3]$	9	95.83%	$2^{40.32}$	$2^{34.0}$
(L11)	$[5,4]$	17	0.83%	$2^{43.58}$	$2^{35.6}$
(L12)	$[5,5]$	19	0.0%	–	–
(L13)	$[5,5]$	16	0.0%	–	–
(L14)	$[5,5]$	9	82.5%	$2^{41.43}$	$2^{34.95}$
(L15)	$[5,5]$	19	0.0%	–	–
(L16)	$[5,5]$	17	0.0%	–	–
(L17)	$[8,2]$	5	100.0%	$2^{35.53}$	$2^{29.17}$
(L18)	$[8,7]$	9	78.75%	$2^{41.56}$	$2^{34.79}$
(L19)	$[8,7]$	5	100.0%	$2^{35.41}$	$2^{29.42}$
(L20)	$[8,8]$	8	79.17%	$2^{41.59}$	$2^{35.78}$
(L21)	$[8,8]$	4	100.0%	$2^{34.76}$	$2^{29.29}$

From these experiments, it seems that the depth has a much more relevant impact than the type. Yet, this seems to be quite tailored to our particular attack. Changing the generation of the different masks is generally more impactful, since it can cut down in three the amount of information given to the attacker.

6 CONCLUSION

In this paper, a theoretical blind side channel attack targeting the LFSR of the Elephant algorithm has been presented. Elephant is a good target. First, Elephant is a finalist to the (NIST) competition for lightweight cryptography candidates for authenticated encryption. Moreover, Elephant is an interesting target because the internal LFSR only depends on the secret key. In other words, in the use-case of Elephant, retrieving the encryption key is equivalent to retrieving the initial state of the LFSR.

Our attack is based on the fact that an attacker can retrieve the Hamming weights of the different bytes in the LFSR. The Elephant design, where there exist relations between the different internal states of the LFSR, is an added vulnerability to our attack. In half the cases, the key is retrieved in less than two days.

Different tweaking options have been considered. Going from the most impactful to the least, they are: changing the mask derivation for domain separation; modifying the LFSR, looking at the importance of depth and type.

Future works may include the inclusion of noise in the simulations, or even better performing the attack on an actual implementation.

ACKNOWLEDGMENTS

This research is part of the chair CyberCNI.fr with support of the FEDER development fund of the Brittany region.

REFERENCES

Beierle, C., Biryukov, A., dos Santos, L. C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q., and Biryukov, A. (2019). Schwaemm and esch: lightweight authenticated encryption and hashing using the sparkle permutation family. *NIST round*, 2.

Bernstein, D. J. (1999). How to Stretch Random Functions: The Security of Protected Counter Sums. *J. Cryptol.*

Bertoni, G., Daemen, J., Peeters, M., and van Assche, G. (2011). The Keccak Reference.

Beyne, T., Chen, Y. L., Dobraunig, C., and Mennink, B. (2020). Dumbo, Jumbo, and Delirium: Parallel Authenticated Encryption for the Lightweight Circus. *IACR Transactions on Symmetric Cryptology*.

Beyne, T., Chen, Y. L., Dobraunig, C., and Mennink, B. (2021). Elephant v2. *NIST lightweight competition*.

Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varici, K., and Verbauwhede, I. (2011). Spongent: a Lightweight Hash Function. In *CCryptographic Hardware and Embedded Systems-CHES*. Springer.

Brier, E., Clavier, C., and Olivier, F. (2004). Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems-CHES*. Springer.

Burman, S., Mukhopadhyay, D., and Veezhinathan, K. (2007). LFSR based stream ciphers are vulnerable to power attacks. In *INDOCRYPT*, volume 4859 of *Lecture Notes in Computer Science*, pages 384–392. Springer.

Chakraborty, A., Mazumdar, B., and Mukhopadhyay, D. (2014). Fibonacci LFSR vs. galois LFSR: which is more vulnerable to power attacks? In *SPACE*, volume 8804 of *Lecture Notes in Computer Science*, pages 14–27. Springer.

Chari, S., Rao, J. R., and Rohatgi, P. (2002). Template attacks. In *Cryptographic Hardware and Embedded Systems-CHES*. Springer.

Clavier, C. and Reynaud, L. (2017). Improved blind side-channel analysis by exploitation of joint distributions of leakages. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 24–44. Springer.

Daemen, J., Hoffert, S., Peeters, M., Assche, G. V., and Keer, R. V. (2020). Xoodyak, a lightweight cryptographic scheme.

Dobraunig, C., Eichlseder, M., Mendel, F., and Schläffer, M. (2014). Ascon. *Submission to the CAESAR competition*.

Gierlichs, B., Batina, L., Tuyls, P., and Preneel, B. (2008). Mutual information analysis. In *Cryptographic Hardware and Embedded Systems-CHES*. Springer.

Giraud, C. (2004). DFA on AES. In *Advanced Encryption Standard -AES*. Springer.

Granger, R., Jovanovic, P., Mennink, B., and Neves, S. (2016). Improved Masking for Tweakable Blockciphers with Applications to Authenticated Encryption. In *EUROCRYPT*. Springer.

Handschuh, H. and Heys, H. M. (1998). A timing attack on rc5. In *International Workshop on Selected Areas in Cryptography*. Springer.

Hell, M., Johansson, T., Maximov, A., Meier, W., and Yoshida, H. (2021). Grain-128aead, round 3 tweak and motivation.

Iwata, T., Khairallah, M., Minematsu, K., and Peyrin, T. (2020). Duel of the titans: the romulus and remus families of lightweight aead algorithms. *IACR Transactions on Symmetric Cryptology*.

Joux, A. and Delaunay, P. (2006). Galois LFSR, Embedded Devices and Side Channel Weaknesses. In *INDOCRYPT*, volume 4329 of *Lecture Notes in Computer Science*, pages 436–451. Springer.

Jurecek, M., Bucek, J., and Lórencz, R. (2019). Side-channel attack on the a5/1 stream cipher. In *Euromicro Conference on Digital System Design (DSD)*. IEEE.

Kazmi, A. R., Afzal, M., Amjad, M. F., Abbas, H., and Yang, X. (2017). Algebraic side channel attack on trivium and grain ciphers. *IEEE Access*.

Kocher, P. C., Jaffe, J., and Jun, B. (1999). Differential power analysis. In *Advances in Cryptology - CRYPTO*. Springer.

Le Bouder, H., Lashermes, R., Linge, Y., Thomas, G., and Zie, J. (2016). A Multi-round Side Channel Attack on AES Using Belief Propagation. In *Foundations and Practice of Security*. Springer.

Linge, Y., Dumas, C., and Lambert-Lacroix, S. (2014). Using the joint distributions of a cryptographic function in side channel analysis. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer.

Luykx, A., Preneel, B., Tischhauser, E., and Yasuda, K. (2016). A MAC Mode for Lightweight Block Ciphers. In Peyrin, T., editor, *Fast Software Encryption FSE*. Springer.

Mangard, S. (2002). A simple power-analysis (spa) attack on implementations of the aes key expansion. In *ICISC*. Springer.

NIST (2015). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. *FIPS 202*.

NIST (2018). *Lightweight Cryptography Standardization Process*.

Rechberger, C. and Oswald, E. (2004). Stream ciphers and side-channel analysis. In *In ECRYPT Workshop, SASC-The State of the Art of Stream Ciphers*. Citeseer.

Samwel, N. and Daemen, J. (2017). DPA on hardware implementations of ascon and keyak. In *Proceedings of the Computing Frontiers Conference*. ACM.

Standaert, F.-X. (2010). Introduction to side-channel attacks. In *Secure integrated circuits and systems*. Springer.