Practical Algebraic Side-Channel Attacks against ACORN

Alexandre Adomnicai[1,2][0000−0003−1210−8046], Laurent Masson[1][0000−0002−5835−4616], and Jacques J.A. Fournier[3][0000−0001−8185−892X]

[1] Trusted Objects, Aix-en-Provence, France,
{a.adomnicai, l.masson}@trusted-objects.com
[2] Mines Saint-Étienne, CEA-Tech, Centre CMP, Gardanne, France,
[3] Univ. Grenoble Alpes, CEA-LETI, DSYS, Grenoble, France,
jacques.fournier@cea.fr

Abstract. The authenticated cipher ACORN is one of the two finalists of the CAESAR competition and is intended for lightweight applications. Because such use cases require protection against physical attacks, several works have been undertaken to achieve secure implementations. Although dedicated threshold and masked schemes have been proposed, no practical side-channel attack against ACORN has been published in the literature yet. It has been theoretically demonstrated that ACORN is vulnerable against differential power analysis but the feasibility of the attack has not been validated in a practical manner. This paper details the results obtained when putting the attack into practice against a software implementation running on a 32-bit micro-controller. Especially, these practical results led us to propose two optimizations of the reference attack: one that requires less knowledge of initial vectors and another one that is less prone to errors and requires fewer acquisitions.

Keywords: ACORN, Authenticated encryption, Side-channel attacks

1 Introduction

In January 2013, the competition for authenticated encryption: security, applicability, and robustness (CAESAR) has been launched with the objective to push for the adoption of authenticated encryption schemes that offer advantages over AES-GCM and are suitable for widespread adoption. In March 2018, the finalists for different use cases were announced. Among them, ACORN is still competing for lightweight applications. This category is defined by various criteria such as compactness of the implementation (in software and hardware), a low overhead for short messages and an intrinsic ability to protect against physical attacks. While several studies have been carried out in order to investigate the last point, most of them discuss the susceptibility of ACORN towards fault attacks [19,14,4]. The first work with regards to side-channel attacks has been recently published [5]. In this paper, the authors propose threshold implementations of some CAESAR candidates, including ACORN, in order to compare their ability to integrate countermeasures against differential power analysis (DPA) in hardware. To justify the need of such countermeasures, they apply the non-specific t-test [13] to each unprotected implementation on a Spartan 6 FPGA in order to detect the presence of leakages. Their results show that ACORN seems to be the most leakage resilient candidate in the unprotected setting and has the lowest area when implemented with countermeasures. The second work dealing with side-channel attacks follows the same approach by studying the integration of the masking countermeasure to the finalists ACORN and Ascon in software [1]. This latter also introduces the first theoretical DPA against ACORN but does not provide any practical results. Therefore, the only two available studies on the susceptibility of ACORN towards side-channel analysis only deal with leakage detection and theoretical attacks.

Our contribution. Although leakage assessment methodologies give a good overview of the resilience of an implementation against side-channel attacks, it might not be sufficient to guarantee its security level [16]. Because such statistic tools are not meant to perform a key recovery, it is recommended to run additional tests (*e.g.* DPA) in order to assess the security of an implementation in an accurate manner. However, the only DPA against ACORN reported in the literature has not been validated in practice. To fill the gap, we run the attack described in [1] on a software implementation of ACORN on a 32-bit microcontroller. In addition to bringing information on the effectiveness of the attack and the difficulties that might be encountered in practice, our results allow us to introduce more efficient attack paths.

Outline. The rest of this paper is organised as follows. Section 2 briefly recalls the specification of ACORN and the principle of correlation eletromagnetic analysis. Section 3 recalls the theoretical attack against this algorithm and provides some missing elements in order to put it into practice. Subsequently, Sect. 4 details how the attack was applied in a practical manner and presents the results obtained. Section 5 introduces two optimized variants of the reference attack, each one having its own advantages. Finally, we summarise our main results and provide some perspectives in Sect. 6.

2 Preliminaries

2.1 ACORN

ACORN [18] is a stream cipher based authenticated encryption with associated data (AEAD) algorithm designed by Hongjun Wu. ACORN uses a 128-bit key, a 128-bit initialization vector (IV) and produces a 128-bit authentication tag. Its internal state is 293-bit long and consists of the concatenation of six LFSRs in addition to a 4-bit register, as shown in Fig.1. We note S^i the state after i updates and S_j the j^{th} bit of the state.

ACORN relies on three main functions: an output keystream generation function, a nonlinear feedback function, and a state update function. The keystream

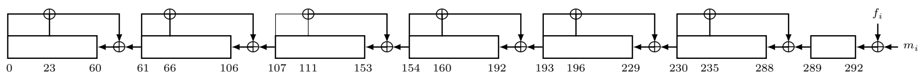


Fig. 1: The concatenation of 6 LFSRs in ACORN. f_i and m_i indicate the overall feedback bit and the message bit for the i^{th} step, respectively.

generation function is defined by

$$\kappa(S) = S_{12} \oplus S_{154} \oplus Maj(S_{235}, S_{61}, S_{193}) \oplus Ch(S_{230}, S_{111}, S_{66}) \tag{1}$$

where $Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$ and $Ch(x, y) = (x \wedge y) \oplus (\neg x \wedge z)$. The nonlinear feedback function is defined by

$$\varphi(S, k, ca, cb) = S_0 \oplus \neg S_{107} \oplus Maj(S_{244}, S_{23}, S_{160}) \oplus (ca \wedge S_{196}) \oplus (cb \wedge k)\,. \tag{2}$$

The variables ca and cb allow to define different variants of the feedback function for the four phases of the cipher: initialization, additional data processing, encryption and tag generation. All of them rely on the state update function, defined in Alg.1, which is the core of ACORN.

Algorithm 1 `StateUpdate`(S^i, m_i, ca, cb)

```
$S^i_{289} \leftarrow S^i_{289} \oplus S^i_{235} \oplus S^i_{230}$        ▷ Update using six LFSRs
$S^i_{230} \leftarrow S^i_{230} \oplus S^i_{196} \oplus S^i_{193}$
$S^i_{193} \leftarrow S^i_{193} \oplus S^i_{160} \oplus S^i_{154}$
$S^i_{154} \leftarrow S^i_{154} \oplus S^i_{111} \oplus S^i_{107}$
$S^i_{107} \leftarrow S^i_{107} \oplus S^i_{66} \oplus S^i_{61}$
$S^i_{61} \leftarrow S^i_{61} \oplus S^i_{23} \oplus S^i_{0}$
$ks_i \leftarrow \kappa(S^i)$
$c_i \leftarrow ks_i \oplus m_i$        ▷ Encryption of the input
$f_i \leftarrow \varphi(S^i, ks_i, ca, cb)$        ▷ Nonlinear feedback bit generation
**for** $j$ from 0 to 291 **do**
    $S^{i+1}_j \leftarrow S^i_{j+1}$        ▷ Shift the state
$S^{i+1}_{292} \leftarrow f_i \oplus m_i$        ▷ Injection of the input
```

Initialization. The initialization phase takes as input the encryption key and the IV. First, the entire state is initialized to zero. Then the cipher is run for 1792 steps as described in Alg.2.

Additional Data Processing. After the initialization step, the associated data is used to update the state. The cipher is run for at least 256 steps, even if there is no associated data to process.

Encryption. At each step of the encryption, one bit from the plaintext is encrypted. The cipher is run for at least 256 steps, even if there is no plaintext to encrypt.

Finalization. At the end, an n-bit authentication tag is computed. The state is updated 768 times and the tag consists of the last n keystream bits generated.

Algorithm 2 `AcornInit`(S^0, K, IV)

```
$(S_0^0, ..., S_{292}^0) \leftarrow (0, ..., 0)$    ▷ Initialize the state to zero
for $i$ from 0 to 127 do
    $S^{i+1} \leftarrow$ StateUpdate$(S^i, K_i, 1, 1)$    ▷ Update the state with key bits as input
for $i$ from 0 to 127 do
    $S^{129+i} \leftarrow$ StateUpdate$(S^{128+i}, IV_i, 1, 1)$    ▷ Update the state with IV bits as input
$S^{257} \leftarrow$ StateUpdate$(S^{256}, K_0 \oplus 1, 1, 1)$
for $i$ from 1 to 1535 do
    $S^{257+i} \leftarrow$ StateUpdate$(S^{256+i}, K_{i \bmod 128}, 1, 1)$    ▷ Update the state with key bits as input
```

2.2 Correlation Electromagnetic Analysis

Since the publication of DPA [9], it is common knowledge that the analysis of the power consumed by the execution of a cryptographic primitive might reveal information about the secret involved. A few years later, correlation power analysis (CPA) has been widely adopted over DPA as it requires fewer traces and has been shown to be more efficient [3]. The principle is to target a sensitive intermediate state of the algorithm which depends on a subpart of the key, and try to predict its value for all hypotheses. The function that defines the intermediate state from the known input and the subkey is called *selection function*. Then, to uncover the link between these predictions and the leakage measurements, the Pearson correlation coefficient between these two variables is computed using an appropriate leakage model. The Hamming weight (HW) and the Hamming distance (HD) models are the most commonly used models to simulate the leakage of a cryptographic device. For each subkey hypothesis, it results in a value between -1 and 1, indicating how much it correlates with the recorded values for every point in time. Finally, the hypothesis which matches with the real subkey should return a significantly higher coefficient than the other hypotheses. The procedure is described in details in Alg. 3. This attack remains valid when analyzing electromagnetic emanations [6,11] instead of power consumption, since they are mainly due to the displacement of current through the rails of the metal layers. In this case, we refer to it as correlation electromagnetic analysis (CEMA).

3 Reference Attack against ACORN

3.1 Theoretical Basics

The attack introduced in [1] details how a DPA can be mounted against leakages caused by the calculation of $S_{292}^{i+1} \leftarrow f_i \oplus m_i$ when updating the state update during the initialization phase for $128 \leq i \leq 255$. More precisely, it assumes the knowledge of the input $m_i = IV_{i-128}$ and thus targets the feedback bits f_i. However, because feedback bits are defined by nonlinear combinations of several

Algorithm 3 $\mathsf{CEMA}(\varphi,\ \mathcal{L},\ D^{1\cdots n},\ [a,b],\ M^{1\cdots n})$

```
Require: Selection function $\varphi$ ; Leakage model $\mathcal{L}$ ; Data acquisitions $D^{1\cdots n}$ ; Interval
    of samples to consider $[a,b]$ ; Input messages $M^{1\cdots n}$
Ensure: subkey candidate $\bar{k}$
  for $i$ from 1 to $n$ do
    for $k$ from 0 to $|\mathcal{K}-1|$ do                ▷ $\mathcal{K}$ denotes the key search space
      $H_k^i \leftarrow \mathcal{L}\left(\varphi(k, M^i)\right)$          ▷ Prediction of the intermediate state leakage
  for $i$ from $a$ to $b$ do                              ▷ For each sample to consider
    for $k$ from 0 to $|\mathcal{K}-1|$ do
      $C_k^i \leftarrow \mathsf{Corr}\left(\left[H_k^1, \cdots, H_k^n\right], \left[D_i^0, \cdots, D_i^n\right]\right)$
  $C_{\bar{k}} \leftarrow \mathsf{max}(C)$                          ▷ Most likely subkey among all samples in $[a,b]$
```

key bits, the attack does not lead to a direct key recovery but returns a system of Boolean equations to be solved. This kind of attack is called algebraic side-channel attack (ASCA) [12] and has already been applied to other stream ciphers such as Trivium and Grain [8].

In the case of ACORN, the state is first updated 128 times with the key. Especially, after the 128th initialization step, the state is as follows

$$\begin{aligned}
&\left(S_0^{128}, ..., S_{164}^{128}\right) = (0, ..., 0)\\
&\left(S_{165}^{128}, ..., S_{198}^{128}\right) = (\neg K_0, ..., \neg K_{33})\\
&\left(S_{199}^{128}, ..., S_{201}^{128}\right) = (K_{34} \oplus K_0, ..., K_{36} \oplus K_2)\\
&\left(S_{202}^{128}, ..., S_{218}^{128}\right) = (\neg K_{37} \oplus K_3 \oplus K_0, ..., \neg K_{53} \oplus K_{19} \oplus K_{16})\\
&\left(S_{219}^{128}, ..., S_{223}^{128}\right) = (K_{54} \oplus K_{20} \oplus K_{17} \oplus K_0, ..., K_{58} \oplus K_{24} \oplus K_{21} \oplus K_4)\\
&\left(S_{224}^{128}, ..., S_{229}^{128}\right) = (\neg K_{59} \oplus K_{25} \oplus K_{22} \oplus K_5 \oplus K_0, ..., \neg K_{64} \oplus K_{30} \oplus K_{27} \oplus K_{10} \oplus K_5)\\
&\left(S_{230}^{128}, ..., S_{261}^{128}\right) = (\neg K_{65} \oplus K_{11} \oplus K_6, ..., \neg K_{96} \oplus K_{42} \oplus K_{37})\\
&\left(S_{262}^{128}, ..., S_{272}^{128}\right) = (K_{97} \oplus K_{43} \oplus K_{38} \oplus f_{97}, ..., K_{107} \oplus K_{53} \oplus K_{48} \oplus f_{107})\\
&\left(S_{273}^{128}, ..., S_{288}^{128}\right) = (\neg K_{108} \oplus K_{54} \oplus K_{49} \oplus K_0 \oplus f_{108}, ..., \neg K_{123} \oplus K_{69} \oplus K_{64} \oplus K_{15} \oplus f_{123})\\
&\left(S_{289}^{128}, ..., S_{292}^{128}\right) = (\neg K_{124} \oplus f_{124}, ..., \neg K_{127} \oplus f_{127})
\end{aligned} \tag{3}$$

where f_i defines the nonlinear feedback bit.

$$f_i = \begin{cases} 1 & \text{if } 0 \le i \le 96\\ K_{i-97} & \text{if } 97 \le i \le 99\\ (\neg K_{i-58}) \wedge (\neg K_{i-100}) \oplus K_{i-97} & \text{if } 100 \le i \le 111\\ (K_{i-58} \oplus K_{i-112}) \wedge (\neg K_{i-100}) \oplus K_{i-97} & \text{if } 112 \le i \le 116\\ \neg\left(K_{i-58} \oplus K_{i-112} \oplus K_{i-117}\right) \wedge (\neg K_{i-100}) \oplus K_{i-97} & \text{if } 117 \le i \le 127 \end{cases} \tag{4}$$

Then, the state is updated with the IV as input for the next 128 steps. As a result, one can run a DPA by targeting the result of the XOR between IV bits and feedback bits in order to get a system of Boolean equations to be solved.

However, f_i is constant for a given key if and only if $i \leq 176$. Indeed, from $i = 177$, IV bits that have been injected into the internal state have been shifted to such an extent that they are involved in the computation of f_i. Therefore, the use of XOR as selection function is only possible from f_{128} to f_{176}, which results in a Boolean system $\mathscr{F}$ that depends on all key bits.

The task of recovering the key bits from $\mathscr{F}$ can be reduced to a variant of the Boolean satisfiability (SAT) problem, which decides whether a given propositional formula in conjunctive normal form (CNF) is satisfiable. As the CNF derived from $\mathscr{F}$ is satisfiable at least by the encryption key K, the purpose of the attack is to get all of the truth assignments of SAT. Because $\mathscr{F}$ defines a system of 49 equations with 128 unknowns, there are so many solutions that we were not able to determine the number of truth assignments by means of 600 core-hours. In order to reduce the number of solutions, it is possible to extend $\mathscr{F}$ by recovering the next feedback bits using more sophisticated selection functions.

Actually, the DPA should not target the next f_i themselves but their component parts that are IV-independent. The principle is to isolate the key bit combinations from the IV bits so that they can be recovered through a DPA and then be added to the Boolean system. As a result, each DPA against f_i for $i \geq 177$ adds $n + 1$ equations to the Boolean system where n is the number of IV bits involved in the calculation of f_i. The authors computed the value of each equation of $\mathscr{F}$ for a random key and investigated how many equations are necessary to return a single key hypothesis. Their experimentations led to the conclusion that $\mathscr{F}$ has a unique truth assignment only if it results from the leakage of at least the first 82 updates (*i.e.* from f_{128} to f_{209}). Therefore, ACORN is theoretically vulnerable to DPA and it is only necessary to have knowledge of the first 82 IV bits to recover the entire encryption key.

3.2 Remarks and Clarifications

Although [1] clearly exhibits how to proceed in order to isolate the key bit combinations when a single IV bit interferes in the recovery of f_i with $i \geq 177$, the case where multiple IV bits are involved is left to the reader as an exercise. Thanks to the distributive property of AND over XOR, f_i can be rewritten in terms of IV bits as shown in Table 1, where f'_i refers to f_i for the null IV.

Table 1: Intermediate bit $\beta_{i\in\mathcal{I}}$ to consider when running a DPA against $f_{128+i} \oplus IV_i$

$\mathcal{I}$	$\beta_{i\in\mathcal{I}}$
$[0, 48]$	$f_{128+i} \oplus IV_i$
$[49, 57]$	$f'_{128+i} \oplus \left(S_{160}^{128+i} \wedge IV_{i-49}\right) \oplus IV_i$
$[58, 62]$	$f'_{128+i} \oplus \left(S_{160}^{128+i} \wedge IV_{i-49}\right) \oplus \left(S_{193}^{128+i} \wedge IV_{i-58}\right) \oplus IV_i$
$[63, 96]$	$f'_{128+i} \oplus \left(S_{160}^{128+i} \wedge IV_{i-49}\right) \oplus \left(S_{193}^{128+i} \wedge IV_{i-58}\right) \oplus \left(S_{111}^{128+i} \wedge IV_{i-63}\right) \oplus IV_i$

It straightforwardly follows the definition of the selection function $\varphi_{i \in \mathcal{I}}$ to use for a given feedback bit index.

$$\varphi_i : \begin{cases} x \mapsto x \oplus IV_i & \text{if } 0 \leq i \leq 48 \\ (x, y) \mapsto x \oplus (y \wedge IV_{i-49}) \oplus IV_i & \text{if } 49 \leq i \leq 57 \\ (x, y, z) \mapsto x \oplus (y \wedge IV_{i-49}) \oplus (z \wedge IV_{i-58}) \oplus IV_i & \text{if } 58 \leq i \leq 62 \\ (x, y, z, t) \mapsto x \oplus (y \wedge IV_{i-49}) \oplus (z \wedge IV_{i-58}) \oplus (t \wedge IV_{i-63}) \oplus IV_i & \text{if } 63 \leq i \leq 96 \end{cases} \tag{5}$$

Throughout this paper, $\mathscr{F}_{\mathcal{I}}$ refers to the system resulting from the leakage of $\beta_{i \in \mathcal{I}}$. In order to express the values of $\mathscr{F}_{\mathcal{I}}$ in terms of key bits, we implemented a software version of ACORN which operates on strings instead of numeric values (*i.e.* `'a'` $\oplus$ `'b'` $=$ `'a ^ b'`). For instance, $\mathscr{F}_{[0,81]}$ which should return a unique solution according to [1], is defined in Eq. 6.

$$\mathscr{F}_{[0,81]} = \begin{cases} (\neg K_{70} \oplus K_{11} \oplus K_{16}) \wedge \neg K_{28} \oplus K_{31} & = f_{128} \\ \vdots & \vdots \\ (\neg (K_{69} \oplus K_{10} \oplus K_{15}) \wedge \neg K_{27} \oplus K_{30} \oplus K_{127} \oplus K_{68} \oplus K_9 \oplus \cdots) \wedge \cdots & = f_{176} \\ (f_{128} \oplus K_{69} \oplus K_{10} \oplus K_{74} \oplus K_{20}) \wedge S_{160}^{177} \oplus \neg (K_{61} \oplus K_2 \oplus K_7) \oplus \cdots & = f'_{177} \\ \vdots & \vdots \\ (f_{160} \oplus (\neg K_{44} \wedge \neg K_2) \oplus K_5 \oplus K_{102} \oplus K_{43} \oplus K_{48}) \wedge (\neg K_{60} \oplus \cdots) \oplus \cdots & = f'_{209} \\ \neg K_{44} \oplus K_7 \oplus K_{10} \oplus K_5 \oplus K_{11} & = S_{160}^{177} \\ \vdots & \vdots \\ K_{76} \oplus K_{17} \oplus K_{22} \oplus K_{39} \oplus K_2 \oplus K_{42} \oplus K_8 \oplus K_{37} \oplus K_0 \oplus K_3 \oplus \cdots & = S_{160}^{209} \\ \neg K_{86} \oplus K_{27} \oplus K_{32} \oplus K_{49} \oplus K_{12} \oplus K_{14} \oplus K_{52} \oplus K_{15} \oplus K_{18} & = S_{193}^{186} \\ \vdots & \vdots \\ (\neg K_{51} \wedge \neg K_9) \oplus K_{21} \oplus K_{109} \oplus K_{50} \oplus K_{55} \oplus K_1 \oplus K_{72} \oplus K_{13} \oplus \cdots & = S_{193}^{209} \\ \neg K_9 & = S_{111}^{191} \\ \vdots & \vdots \\ \neg K_{27} & = S_{111}^{209} \end{cases} \tag{6}$$

Because the equations resulting from the recovery of S_{111}^{128+i} depend on a single key bit, they are especially useful to solve $\mathscr{F}_{[0,81]}$. As a result, it might be interesting to ignore leakages related to f_{128+i} for $i \in [0, 62]$ and rather focus on $\mathscr{F}_{[63,96]}$.

4 From Theory to Practice

4.1 Targeted Implementation

Although ACORN is designed to process one bit per step, because its smallest LFSR is 37-bit long, up to 37 steps can be processed in parallel. Within the scope of the CAESAR contest, Hongjun Wu provided an optimized software implementation which processes 32 steps at once. In this way, each function defined in Sect. 2 should be seen as operating on 32-bit words instead of bits. Its implementation dedicates a 64-bit register to each LFSR. Although it consumes more memory than needed, since all LFSRs contains less than 64 bits, it increases the performances by saving some instructions in order to build the 32-bit working variables. An ARM assembly implementation of the state update function based on the same principle is provided in [1]. We chose to run the attack against this specific implementation, as it is the most appropriate for 32-bit platforms. The hard-coded 128-bit encryption key $K =$ `'Encryption key K'` was used to encrypt and authenticate 5 000 messages, using random IVs.

4.2 Experimental Setup

All practical experiments presented below were done using a microcontroller equipped with an ARM Cortex-M3 running at 24MHz. Note that the device under test does not embed any hardware countermeasure against side-channel attacks. A trigger signal was inserted at the beginning and the end of the initialization phase in order to guarantee a proper synchronization. EM emanations were measured using a Langer HF-U 5 near-field probe (30 MHz - 3 GHz) combined with a Langer PA 303 BNC preamplifier providing a gain of 30dB. The sampling acquisition was performed using a PicoScope 6404D sampled at 1GS/s. We recorded the leakage from state updates where IV words are given as input, but also from five further ones in case they would also contain information to exploit. As shown in Fig. 2, state updates are clearly discernible and each of them are roughly made up of 10 000 samples.

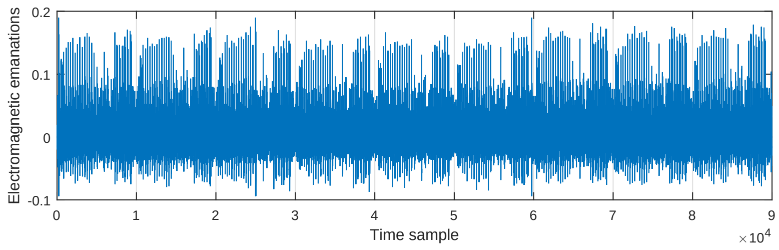


Fig. 2: Data acquisition of nine 32-bit state updates during the initialization phase

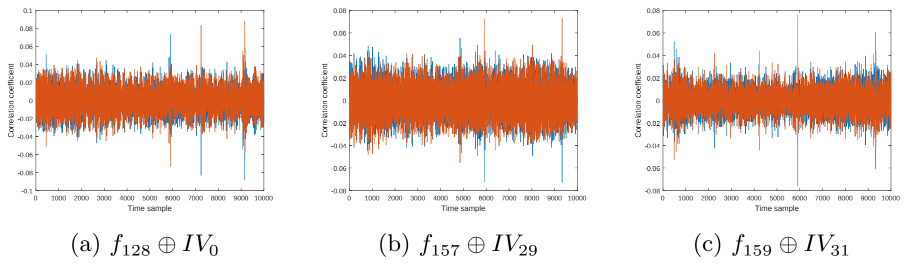
4.3 Practical Correlation Electromagnetic Analysis

Even if the targeted feedback bits are actually stored in 32-bit registers, it has been proven that one can compute a partial correlation of the entire variable in order to reduce the computational complexity [17]. Therefore we chose to apply the attack as defined above, in a mono-bit manner, using the Hamming weight leakage model and the Pearson's correlation coefficient as distinguisher. In order to precisely target leakages related to the insertion of f_{128+i} into the state, the window on which the attack is run depends on the feedback bit index. More precisely, an attack against $f_{128+i} \oplus IV_i$ is managed by executing

$$\mathsf{CEMA}\left(\varphi_i, \mathrm{HW}, D^{1\cdots 5\,000}, \left[10\,000 \times \left\lceil \frac{i}{32} \right\rceil + 1, 10\,000 \times \left\lceil \frac{i+32}{32} \right\rceil\right], IV^{1\cdots 5\,000}\right).$$

As suggested in the theoretical specification, we run the attack for i from 0 to 81. After assigning the CEMA results to the corresponding equations within $\mathscr{F}_{[0,81]}$, they are converted into CNF formulas using the `bc2cnf` tool [7] and finally given as input to the SAT solver `CryptoMiniSat5` [15]. On the first try, it turns out that the input system is not satisfiable and therefore does not lead to a key recovery. Because this issue can be due to many factors (*e.g.* some erroneous CEMA results or ineffectiveness of some selection functions), we carried out investigations starting by visually examining the CEMA output for various feedback bits.

Figure 3 illustrates the points of interests (POI) for some of them. The first observation that can be made is that information leakage is not identical for all feedback bits. For instance, Fig. 3a shows three samples (5 913, 7 245 and 9 160) that might reveal information about f_{128} while Fig. 3b shows only two (5 913 and 9 329) regarding f_{157}.



(a) $f_{128} \oplus IV_0$ (b) $f_{157} \oplus IV_{29}$ (c) $f_{159} \oplus IV_{31}$

Fig. 3: POI for several feedback bits during the first state update

Because the implementation processes 32 steps at once, all but the last four – stored in the 4-bit register – feedback bits have already been updated using S_{230} and S_{235} before being added to the state. For instance, after the state update of S^{128} with $IV_{0\cdots31}$ as input, the last 32 bits of S^{160} are as follows.

$$\begin{aligned}\left(S^{160}_{261}, \cdots, S^{160}_{288}\right) &= \left(f_{128} \oplus S^{132}_{230} \oplus S^{132}_{235} \oplus IV_0, \cdots, f_{155} \oplus S^{159}_{230} \oplus S^{159}_{235} \oplus IV_{27}\right) \\ \left(S^{160}_{289}, \cdots, S^{160}_{292}\right) &= \left(f_{156} \oplus IV_{28}, \cdots, f_{159} \oplus IV_{31}\right)\end{aligned} \tag{7}$$

Especially, the implementation computes $(f_i \parallel \cdots \parallel f_{i+31}) \oplus (IV_i \parallel \cdots \parallel IV_{i+31})$ before finally updating its 28 most significant bits and adding it into the state. Therefore, $\varphi_{i\in[0,48]}$ should not only return a candidate for f_{128+i}, but also for $f^1_{128+i} = f_{128+i} \oplus S^{132+i}_{230} \oplus S^{132+i}_{235}$ when $i < 28 \bmod 32$. Moreover, the implementation of the state update function is generic in the sense that the input is always encrypted using the keystream, even during the initialization phase. Although encryption is not necessary during this phase, it allows the use of the same code through all the authenticated encryption process. As a result, the selection function $\varphi_{i\in[0,48]}$ also targets $ks_i \oplus IV_i$ unintentionally.

These remarks highlight the first difficulty when putting the attack into practice. Because selection functions can lead to the recovery of several key bit combinations (keystream, feedback and updated feedback bits), an attacker has to associate each leakage to an intermediate value. Indeed, if the highest correlation coefficient is reached for the keystream bit but its value is assigned to the feedback bit equation, then the ASCA will fail because of an erroneous Boolean system. For instance, this scenario is depicted in Fig. 3c where the highest correlation peak is reached for the leakage of $ks_{159} \oplus IV_{31}$ instead of $f_{159} \oplus IV_{31}$. The methodology that was used to clearly identify each leakage is described below.

When targeting software implementations on load/store architectures, data transfers due to memory accesses are known to leak the most information compared to arithmetic and logic operations, which only occur between registers and are usually more difficult to exploit in practice [2,10]. Especially, on top of memory accesses that store the last 32 bits into the state as described in Eq. 7, the assembly code under test performs two additional store instructions that are likely to be critical. It consists of $(ks_i \parallel \cdots \parallel ks_{i+31}) \oplus (IV_i \parallel \cdots \parallel IV_{i+31})$ as computation of the ciphertext, and $(f^1_{128+i} \parallel \cdots \parallel f^1_{155+i}) \oplus (IV_i \parallel \cdots \parallel IV_{i+27})$ as computation of the updated feedback word in a temporary register. Therefore, attacks using $\varphi_{i \in [0,48]}$ should lead to three leakages for $i < 28 \bmod 32$ and only two for $i \geq 28 \bmod 32$, which is consistent with the results from Fig. 3.

As a result, our first attempt to run the attack in practice led to an unsatisfiable system because some CEMA results did not match the expected key bit combinations. More generally, our investigations highlight that the theoretical DPA against ACORN as described in Sect. 3 does not necessarily apply to all unprotected implementations. However, a tweaked version of the attack can still be applied in order to deal with exploitable leakages on the device under test. We chose to ignore leakages related to ciphertext computations as they can be easily avoided during the initialization phase.

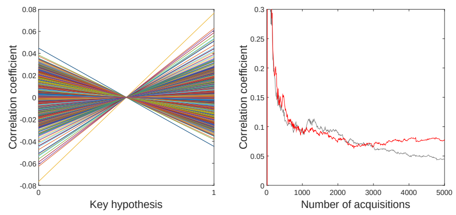
The required modifications affect some of the selection functions. Indeed, even if they remain valid when $i \geq 28 \bmod 32$, this is not the case anymore from $i = 54$ since S^{132+i}_{235} depends on IV_0. In this case, additional IV bits have to be considered. The tweaked selection functions are noted φ^1_i and are defined in Eq. 8.

$$\varphi^1_i = \begin{cases} \varphi_i & \text{if } i \geq 28 \bmod 32 \text{ or } i \leq 54 \\ \varphi_{i-54} \circ \varphi_i & \text{if } 54 \leq i \leq 58 \\ \varphi_{i-59} \circ \varphi_{i-54} \circ \varphi_i & \text{otherwise} \end{cases} \tag{8}$$

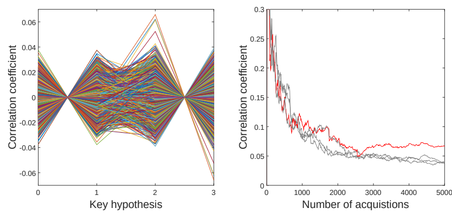
Of course, the Boolean system $\mathscr{F}$ has to be modified in order to be compliant with the intermediate bits defined by φ^1_i, and we refer to this variant as $\mathscr{F}^1$. As leakages related to keystream words are not taken into consideration, the attack is run on the last 3 000 samples of each state update window, by executing for i from 0 to 81,

$$\mathsf{CEMA}\left(\varphi^1_i, \mathrm{HW}, D^{1\cdots 5\,000}, \left[10\,000 \times \left\lceil \frac{i}{32} \right\rceil + 7\,000, 10\,000 \times \left\lceil \frac{i+32}{32} \right\rceil\right], IV^{1\cdots 5\,000}\right).$$

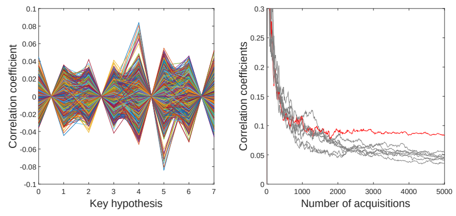
This time, the attack is successful as the resulting system $\mathscr{F}^1_{[0,81]}$ is satisfiable and returns the expected key as the unique solution. Moreover, unlike $\mathscr{F}_{[0,n]}$ that requires $n \geq 81$ to return a unique solution, $n \geq 78$ is enough for $\mathscr{F}^1_{[0,n]}$. For each kind of selection function, Fig. 4 shows a CEMA result and the maximum correlation coefficient reached for each key hypothesis, depending on the number of acquisitions.



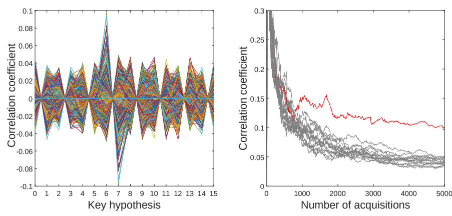
(a) φ^1_0 returns $f^1_{128} = 1$



(b) φ^1_{49} returns $f^1_{177} \parallel S^{177}_{160} = (10)_2$



(c) φ^1_{58} returns $f^1_{186} \parallel S^{186}_{160} \parallel S^{186}_{193} = (100)_2$



(d) φ_{63} returns $f_{191}||S^{191}_{160}||S^{191}_{193}||S^{191}_{111} = (0110)_2$

Fig. 4: Experimental results for the different selection functions

4.4 Lessons Learned

Several conclusions can be drawn from these experimentations. First, although the practical application does not exactly follows the theoretical specification, it still validates the reference attack in the sense that the evaluation of selection functions remains the same. Indeed, the XOR of additional IV bits does not change the way the hypothetical bits interact with each other as detailed in Eq. 9.

$$\begin{aligned}
\varphi_{65}(x,y,z,t) &= x \oplus (y \wedge IV_{16}) \oplus (z \oplus IV_7) \oplus (t \wedge IV_2) \oplus IV_{65} \\
\varphi^1_{65}(x,y,z,t) &= x \oplus (y \wedge IV_{16}) \oplus (z \oplus IV_7) \oplus (t \wedge IV_2) \oplus (IV_{65} \oplus IV_{11} \oplus IV_6)
\end{aligned} \tag{9}$$

Second, since f_{128+i} has to be XORed with IV_i at some point, it might be tempting to perform this calculation during the encryption step in order to achieve a generic implementation of the state update function. However, as shown by our practical experiments, it can lead to additional leakages that an attacker could exploit. Therefore, we argue that the encryption computation should be removed during the initialization phase and that the XOR with IV_i should only occur once f_{128+i} has been entirely computed.

Third, some of the selection functions can be used to recover several intermediate values during the initialization phase (*e.g.* keystream and feedback bits for $\varphi_{i\in[0,48]}$). On the one hand, it introduces the fact that several variants of the attack can be defined depending on the leakage available to the attacker. On the other hand, it requires to clearly identify the points of interests in order to avoid misinterpretation of the results. Finally, some of the selection functions perform better than others. In the next section, we discuss how to take advantage of these results in order to propose more efficient attack paths.

5 Other Attack Variants

5.1 Minimizing the Knowledge of Initial Vectors

Because the knowledge of plaintexts (or IVs in the case of ACORN) is sometimes an unrealistic assumption in practice, it might be interesting to identify the most efficient attack path given the fewer input bits to consider.

In this case, focusing on $\varphi_{i\in[0,48]}$ is of great interest as knowledge of a single IV bit allows to recover several key bit combinations. Because ACORN is defined by the concatenation of six LFSRs, each feedback bit is updated six times before being thrown from the internal state.Therefore, regardless of a potential leakage related to the keystream computation, $\varphi_{i\in[0,48]}$ could theoretically be used to target up to seven key bit combinations: the feedback bit itself f_{128+i} and its six updated values, noted from f^1_{128+i} to f^6_{128+i} and defined in Eq. 10, which are computed just before being shifted in each LFSR.

$$\begin{aligned}
f_i^1 &= f_i \oplus S_{235}^{i+4} \oplus S_{230}^{i+4}\\
f_i^2 &= f_i^1 \oplus S_{196}^{i+63} \oplus S_{193}^{i+63}\\
f_i^3 &= f_i^2 \oplus S_{160}^{i+100} \oplus S_{154}^{i+100}\\
f_i^4 &= f_i^3 \oplus S_{111}^{i+139} \oplus S_{107}^{i+139}\\
f_i^5 &= f_i^4 \oplus S_{66}^{i+186} \oplus S_{61}^{i+186}\\
f_i^6 &= f_i^5 \oplus S_{23}^{i+232} \oplus S_{0}^{i+232}
\end{aligned} \tag{10}$$

In order to investigate whether this statement is verified in practice, we ran attacks using $\varphi_{i\in[0,48]}$ on the same acquisitions but this time, on the entire window of 90 000 samples. Indeed, focusing on the state updates with IV as input only allows to exploit leakages related to the update of three LFSRs. Therefore, also considering five further 32-bit state updates gives access to leakages of all LFSRs' updates. As shown in Fig. 5, each state update leads to several leakages in time (usually two). In addition to the leakage produced by the final store instruction, we suspect that the other peak is due to a previous memory access

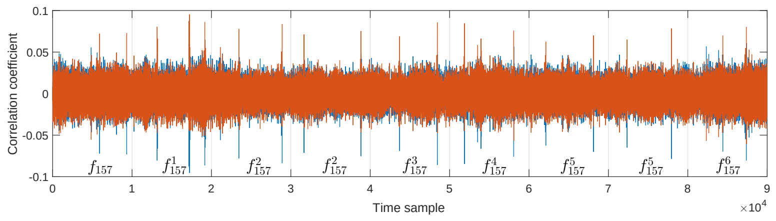


Fig. 5: Leakage in time from $\mathsf{CEMA}\left(\varphi_{29}, \mathrm{HW}, D^{1\cdots 5\,000}, [1, 90\,000], IV^{1\cdots 5\,000}\right)$

that loads the state from RAM to registers. Note that a 32-bit shift does not necessarily imply an LFSR update and thus, several leakages may refer to the same intermediate value. Therefore, the window to consider when targeting a specific feedback bit depends on its index. For instance, Fig. 5 indicates for each state update the key bit combination targeted by φ_{29}.

We tried to apply this attack to our acquisitions. Because the device under test does not leak $f_{128+i} \oplus IV_i$ when $i < 28 \bmod 32$, we were able to exploit six leakages in this case and seven otherwise. The resulting Boolean system is noted $\mathscr{F}^{1\to 6}_{[0,n]}$ where n refers to the number of IV bits considered. Finally, the SAT solver returned the correct key hypothesis as the unique truth assignment for $n \geq 18$. However, the resulting system required more than one hour to be solved on a commonly available laptop while previous systems required less than a second. Indeed, the more an intermediate value is updated, the more terms are involved in its definition, significantly increasing the number of CNF clauses in the resulting system.

5.2 Maximizing the Practical Efficiency

In cases where the attacker has full knowledge of the IVs, other leakages should be preferred. As mentioned in Sect. 3.2, leakages related to f_{128+i} for $i \in [63, 96]$ are of particular interest as some components targeted by $\varphi_{i\in[63,96]}$ refer to single key bits, not combinations of them, allowing to simplify the Boolean system. Moreover, focusing on $\mathscr{F}_{[63,96]}$ brings additional benefits from a practical point of view.

First, it results from Fig. 4 that φ_{64} shows better results than the other selection functions. Especially, it seems that the more IV bits involved in the selection function, the more efficient it is. This can be explained by the fact that selection functions make IV bits interact with hypotheses through the bit-wise AND operator, which is nonlinear. Nonlinearity is a valuable property for selection functions as it ensures a good distinguishability between the correct and incorrect hypotheses and reduces the risk of false positives in practice. As a result, $\varphi_{i\in[63,96]}$ requires fewer acquisitions than other selection functions for the correct hypothesis to stand out.

Second, attacking an intermediate bit using $\varphi_{i\in[63,96]}$ returns a result for four Boolean equations at once. This allows to build a meaningful system by targeting fewer intermediate values and thus, making this attack path less prone to errors. For instance, $\mathscr{F}_{[64,95]}$ is composed of $32\times4 = 128$ equations and has only six truth assignments. Another benefit from this variant is the fact that all the leakages take place during the same state update. Therefore, it is of great interest for 32-bit implementations since it does not require to carefully choose the window to attack given the index of the targeted feedback bit. We ran this attack on our acquisitions, using still $\varphi^1_{i\in[64,95]}$ to be compliant with the implementation under test. Solving $\mathscr{F}^1_{[64,95]}$ led to the correct key as the only solution. In order to highlight all the differences and subtleties between the different attack paths discussed above, Table 2 summarizes all the practical results reported in this paper.

Table 2: Summary and comparison of our practical experiments

	$\mathscr{F}^1_{[0,78]}$	$\mathscr{F}^1_{[64,95]}$	$\mathscr{F}^{1\to6}_{[0,18]}$
IV bits to consider	$IV_{i\in[0,78]}$	$IV_{i\in[0,95]}$	$IV_{i\in\mathbf{[0,18]}}$
# of required acquisitions	$\geq 4\,000$	$\geq \mathbf{2\,000}$	$\geq 4\,000$
# of attacked bits	79	**32**	114
# of equations	148	128	**114**
# of CNF clauses	2165	**1804**	4251
Solving time (i5-6200U CPU)	0.05sec	**0.04sec**	87min17sec

6 Conclusion and Perspectives

The main objective of this paper was to validate the practical feasibility of side-channel attacks against ACORN. To do so, we first defined all selection functions required to put the attack introduced in [1] into practice. Because our experimental setup did not allow us to exploit some leakages required by the theoretical specification, we had to make some minor changes in the selection functions and thus in the resulting Boolean system, in order to achieve a successful attack against the 32-bit implementation under test. However, it does not call into question the reference attack as the results of the selection functions' evaluation remain valid in both cases. Among the different observations made during our experimentations, two of them allowed us to propose optimized variants of the attack. First, one of the selection functions can actually be used to recover several intermediate values, not just the feedback bit itself. It led to an attack that minimizes the number of IV bits to consider. On the device under test, we were able to recover the encryption key with only knowledge of 19 IV bits. Second, another selection function shows significantly better results as it provides a higher distinguishability of the correct hypothesis for fewer acquisitions. This observation led to an attack path that requires to target fewer intermediate values, and is therefore is less prone to errors. On the device under test, we were able to recover the encryption key by targetting 32 intermediate values with half as many acquisitions than other attack paths.

Further work should be undertaken on protected implementations in order to determine whether the selection functions discussed in this paper are efficient enough to deal with high-order side channel analyses. Moreover, the integration of countermeasures such as hiding and shuffling in the specific case of ACORN has not been studied yet and could be of great benefit as some selection functions require to clearly identify specific points of interests.

References

1. Adomnicai, A., Fournier, J.J., Masson, L.: Masking the Lightweight Authenticated Ciphers ACORN and Ascon in Software. In: Tiplea, F.L., Warinschi, B. (eds.) Cryptography and Information Security in the Balkans. Springer International Publishing, Cham (2018), `https://eprint.iacr.org/2018/708`
2. Biryukov, A., Dinu, D., Großschädl, J.: Correlation Power Analysis of Lightweight Block Ciphers: From Theory to Practice, pp. 537–557. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_29
3. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model, pp. 16–29. Springer Berlin Heidelberg, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
4. Dey, P., Rohit, R.S., Adhikari, A.: Full Key Recovery of ACORN with a Single Fault. J. Inf. Secur. Appl. **29**(C), 57–64 (Aug 2016). https://doi.org/10.1016/j.jisa.2016.03.003
5. Diehl, W., Abdulgadir, A., Farahmand, F., Kaps, J.P., Gaj, K.: Comparison of cost of protection against differential power analysis of selected authenticated ciphers. In: 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 147–152 (April 2018). https://doi.org/10.1109/HST.2018.8383904
6. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Analysis: Concrete Results, pp. 251–261. Springer Berlin Heidelberg, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44709-1_21
7. Junttila, T.A., Niemelä, I.: Towards an Efficient Tableau Method for Boolean Circuit Satisfiability Checking. In: Lloyd, J., Dahl, V., Furbach, U., Kerber, M., Lau, K.K., Palamidessi, C., Pereira, L.M., Sagiv, Y., Stuckey, P.J. (eds.) Computational Logic — CL 2000. pp. 553–567. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
8. Kazmi, A.R., Afzal, M., Amjad, M.F., Abbas, H., Yang, X.: Algebraic Side Channel Attack on Trivium and Grain Ciphers. IEEE Access **5**, 23958–23968 (2017). https://doi.org/10.1109/ACCESS.2017.2766234
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology.

pp. 388–397. CRYPTO '99, Springer-Verlag, London, UK, UK (1999), `http://dl.acm.org/citation.cfm?id=646764.703989`

10. McCann, D., Eder, K., Oswald, E.: Characterising and Comparing the Energy Consumption of Side Channel Attack Countermeasures and Lightweight Cryptography on Embedded Devices. In: 2015 International Workshop on Secure Internet of Things (SIoT). pp. 65–71 (Sept 2015). https://doi.org/10.1109/SIOT.2015.11
11. Quisquater, J.J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards, pp. 200–210. Springer Berlin Heidelberg, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-45418-7_17
12. Renauld, M., Standaert, F.X.: Algebraic Side-Channel Attacks. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) Information Security and Cryptology. pp. 393–410. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
13. Schneider, T., Moradi, A.: Leakage Assessment Methodology. Journal of Cryptographic Engineering **6**(2), 85–99 (Jun 2016). https://doi.org/10.1007/s13389-016-0120-y
14. Siddhanti, A., Sarkar, S., Maitra, S., Chattopadhyay, A.: Differential Fault Attack on Grain v1, ACORN v3 and Lizard. In: SPACE (2017)
15. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT Solvers to Cryptographic Problems. In: Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings. pp. 244–257 (2009). https://doi.org/10.1007/978-3-642-02777-2_24
16. Standaert, F.X.: How (not) to Use Welch's T-test in Side-Channel Security Evaluations. Cryptology ePrint Archive, Report 2017/138 (2017), `https://eprint.iacr.org/2017/138`
17. Tunstall, M., Hanley, N., McEvoy, R., Whelan, C., Murphy, C., Marnane, W.: Correlation Power Analysis of Large Word Sizes (2007), `http://www.geocities.ws/mike.tunstall/papers/THMWMM.pdf`
18. Wu, H.: ACORN: A Lightweight Authenticated Cipher (v3). Submission to the CAESAR competition: `https://competitions.cr.yp.to/round3/acornv3.pdf` (2016)
19. Zhang, X., Feng, X., Lin, D.: Fault Attack on ACORN v3. The Computer Journal (2018). https://doi.org/10.1093/comjnl/bxy044