# Fixing the CLOC with Fine-grain Leakage Analysis

William Diehl
Virginia Tech
Blacksburg, Virginia
wdiehl@vt.edu

Farnoud Farahmand
George Mason University
Fairfax, Virginia
ffarahma@gmu.edu

Abubakr Abdulgadir
George Mason University
Fairfax, Virginia
aabdulga@gmu.edu

Jens-Peter Kaps
George Mason University
Fairfax, Virginia
jkaps@gmu.edu

Kris Gaj
George Mason University
Fairfax, Virginia
kgaj@gmu.edu

## ABSTRACT

Authenticated ciphers offer the promise of improved security for resource-constrained devices. Recent cryptographic contests and standardization efforts are evaluating authenticated ciphers for performance and security, including resistance to Differential Power Analysis (DPA). In this research, we study the CLOC-AES authenticated cipher in terms of vulnerability to DPA and cost of implementation of countermeasures against DPA. Using the FOBOS test architecture, we first show that an FPGA implementation of CLOC is vulnerable to DPA through Test Vector Leakage Assessment methodology (i.e., t-tests). After applying DPA countermeasures, we show that protected CLOC implementations pass t-tests, except for discrete leakage corresponding to a data-dependent branch condition in the CLOC specification. Using an enhanced tool called FOBOS Profiler, we analyze the source of t-test failure down to the exact clock cycle and device state, to confirm the source of leakage. We introduce a new protected non-linear transformation into the datapath, remove all data-dependent decision criteria from the device controller, and verify that the updated protected implementations pass t-tests. We show that the cost of including the protected non-linear transformation leads to 3.8 factor growth in area, 48% reduction in throughput, and 86% reduction in throughput-to-area ratio, compared to the unprotected implementation. Our analysis shows the high cost of DPA-protected non-linear transformations in authenticated ciphers above the cryptographic primitive layer.

## CCS CONCEPTS

• **Security and privacy → Cryptography**; • **Computer systems organization →** *Embedded and cyber-physical systems*;

## KEYWORDS

Cryptography; authenticated cipher; FPGA, DPA; t-test; power

**ACM Reference format:**
William Diehl, Farnoud Farahmand, Abubakr Abdulgadir, Jens-Peter Kaps, and Kris Gaj. 2018. Fixing the CLOC with Fine-grain Leakage Analysis. In *Proceedings of The Second Workshop on Attacks and Solutions in Hardware Security, Toronto, ON, Canada, October 19, 2018 (ASHES'18),* 6 pages.
https://doi.org/10.1145/3266444.3266450

## 1 INTRODUCTION

Resource-constrained applications in the Internet of Things, e.g., automotive, home security, smart grid, and medical applications, require cryptographic security to protect sensitive data. Since authenticated ciphers combine the cryptographic services of confidentiality, integrity, and authentication into one algorithmic construct, they offer potential resource savings and improvements in security, compared to the use of separate cryptographic algorithms (e.g., block ciphers and hashes) required to provide these services.

Current cryptographic algorithms, which have either been subjected to extensive public scrutiny or standardized in Federal Information Processing Standards (FIPS), are generally secure against cryptanalysis or brute force attacks. However, they are implemented in physically imperfect hardware and software, and can leak sensitive information. Using so-called side-channel attacks (SCA), an adversary can monitor physical phenomena associated with data emanating from the device during operation, such as a power signature or electromagnetic radiation, and often deduce the contents of sensitive variables.

Differential Power Analysis (DPA), a powerful SCA pioneered in [14] and expanded in [15], enables an attacker to monitor known inputs and outputs of a cryptographic algorithm, collect power signatures of the device in operation (e.g., traces), and attempt to correlate differential power to the contents of a sensitive variable (such as a secret key) according to a hypothetical power model.

Recent cryptographic competitions and standardization efforts, such as the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR), and the upcoming NIST lightweight cryptography standardization effort, seek to evaluate authenticated ciphers according to several criteria, including resistance to SCA, and ease of inclusion of countermeasures against SCA [1, 3, 19]. In support of this effort, we analyze a recent CAESAR-candidate authenticated cipher, CLOC-AES, to determine its cost of protection against DPA. Using register transfer level (RTL) methodology, we first implement the cipher in the Spartan-6 FPGA, and demonstrate susceptibility to information leakage using Test Vector

Leakage Assessment (TVLA) methodology, or t-test [9]. Evaluations are performed using the Flexible Open-source workBench fOr Side-channel analysis (FOBOS) test architecture [5]. We then protect the cipher against DPA using Threshold Implementation (TI) countermeasures [17], and verify improved resistance to DPA using a FOBOS t-test.

After our first attempt to protect the cipher, we note that several areas of discrete leakage remain. We address the difficult task of locating discrete leakage through the introduction of a new tool, FOBOS Profiler, which correlates time-domain power samples to exact clock cycle and device state, in order to determine the source of leakage. Using FOBOS Profiler and other diagnostic techniques, we are able to confirm the source of leakage as a data-dependent branch condition inherent to the CLOC specification, and correct this source of leakage through protection of a modified non-linear transformation in the authenticated cipher layer of the algorithm.

Our contributions in this research are twofold. First, we demonstrate an efficient method for fine-grain leakage analysis of cryptographic platforms, available to the public as an open-source tool without requiring expensive equipment. Second, we identify an issue not previously addressed in literature – the quantification of the DPA protection costs for significant non-linear transformations in authenticated ciphers which are above the cryptographic primitive (e.g., AES).

## 2 BACKGROUND

### 2.1 Authenticated Ciphers

Authenticated ciphers ensure confidentiality, integrity, and authentication by combining a cipher with a keyed-hash function. Inputs consist of *Message*, such as *Plaintext* or *Ciphertext*, Associated Data (*AD*), which is not encrypted but used for *Tag* generation, a public message number (*Npub*) such a nonce (i.e., number used once), and a secret *Key*. The outputs of authenticated encryptions include *Ciphertext* and *Tag*, which is a function of all input values. During authenticated decryption, the user must supply *Ciphertext*, as well as the original *Npub*, *AD*, *Key*, and *Tag*. The cipher computes *Tag'* based on input values in a step called tag verification. If *Tag* = *Tag'*, then the decrypted *Plaintext* is released.

### 2.2 CLOC-AES

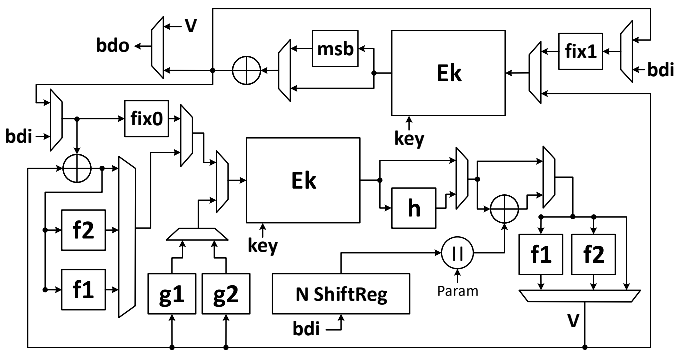

**Figure 1: CLOC top-level datapath, where $E_k$ denotes an AES core. All buses are 128 bits unless noted.**

CLOC, or Compact Low-Overhead Counter Feedback Mode (CFB), is a block cipher mode of operation for authenticated encryption described in [13]. The CLOC top-level datapath of our implementation is shown in Fig. 1.

CLOC can use the AES (Advanced Encryption Standard) or TWINE block ciphers as underlying cryptographic primitives; we use AES in this research. AES, which is the U.S. federal standard and a de-facto worldwide standard for symmetric block ciphers as defined in [18], uses a 128-bit key, encrypts (or decrypts) 128-bit blocks of plaintext (or ciphertext), and consists of 10 rounds. AES consists of four transformations which update the state: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

Our AES implementation is based on [2, 7, 16], and is designed for efficient protection against DPA using threshold implementation (TI) countermeasures. This implementation uses an 8-bit internal datapath, and computes one 128-bit block of ciphertext in 205 clock cycles using a 5-stage pipelined round function. The methods used to implement TI countermeasures are described subsequently. Additionally, this AES implementation performs only encryption (not decryption), since decryption is not required by the CLOC specification.

According to [13], two AES encryptions must be performed in each round of CLOC – one for encryption or decryption, and one for tag generation. We use the implementation of CLOC-AES, available at [8], where two AES cores are used in parallel, in order to reduce latency of each round of CLOC to the equivalent latency of one AES encryption. However, we substitute the AES cores in [8] with the 8-bit pipelined version described above.

Although the AES cores use 8-bit internal bus widths, the datapath for the CLOC authenticated cipher layer remains 128 bits. CLOC receives input data in 128-bit words through the `bdi` port, and processes each 128-bit block in 206 clock cycles. $f1$, $f2$, $g1$, $g2$ and $h$ are the authenticated cipher-layer functions used in CLOC, shown in Fig. 1, $Key$, $Npub$ and $Tag$ sizes are 128, 96 and 64 bits, respectively, which are the parameters recommended in [13].

Our CLOC-AES implementation is fully compliant with the CAESAR Hardware Applications Programming Interface (API) for Authenticated Ciphers (documented at [11]), and uses I/O and formatting utilities of the CAESAR Hardware Development Package, available at [6] and described in [12].

### 2.3 Previous Research

There have been extensive investigations of DPA countermeasures for block ciphers, including AES (for example, [2, 16]), but relatively few reports of extension of countermeasures for authenticated ciphers. Some discussion of DPA countermeasures for authenticated ciphers, including Ascon and Keyak, is provided in [10, 20]. However, these reports essentially deal with protection of one or more features of the cryptographic primitives of these ciphers (e.g., non-linear substitution layers), and are not concerned with countermeasures above the primitive layer. Additionally, the authors in [7] conduct a comparative study of costs of protecting multiple CAESAR-candidate authenticated ciphers against DPA. However, the effect of large non-linear transformations at the authenticated cipher layer on the final protection costs of CAESAR-candidate ciphers is not discussed in any of the above sources.

## 3 METHODOLOGY

### 3.1 Threshold Implementations

To protect CLOC-AES against DPA, we choose an algorithmic countermeasure called Threshold Implementations (TI) [17]. In TI, sensitive data is separated into shares, on which computations are performed independently. TI provide DPA security in the presence of glitches, provided that they satisfy the properties of non-completeness (i.e., each share must lack at least one piece of sensitive data), correctness (i.e., the final recombination of the result must be correct), and uniformity (i.e., an output distribution should match the input distribution).

Achieving TI which are both non-complete and uniform is often challenging. We apply a typical method for ensuring uniformity, which is to refresh TI shares after non-linear transformations using additional randomness. Refresh randomness is supplied by a Pseudo Random Number Generator (PRNG) embedded in the protected implementation of the cipher.

We use a hybrid 2-share / 3-share approach, where all of the linear transformations in each cipher are protected using two shares, and expand to three shares only for non-linear transformations [2]. We compress our three shares to two shares following each non-linear operation. This increases our randomness requirements, as we are required to provide for both "resharing" and mask "refreshing," but reduces total resource usage, since less total logic is required.

### 3.2 Protection of CLOC-AES against DPA

The AES cryptographic primitive is protected using methods described in [2, 7, 16]. We use the Tower Fields method to more efficiently compute subfields of GF($2^8$) [4]. Since even one TI-protected S-Box is costly, we construct only one 3-share TI-protected 8-bit inverter; all other transformations, which are linear, use only two shares. The 2- / 3-share hybrid construction requires 16 bits of fresh randomness for resharing from two to three shares, and 24 fresh remasking bits, for a total of 40 random bits per clock cycle.

The authenticated cipher layer is easier to protect than the cryptographic primitive. In general, both the public and secret data arriving through the external interface are separated into two shares. However, several low-cost 3-share TI are necessary to protect non-linear functions such as one-zero padding (*ozp*).

### 3.3 Test Vector Leakage Assessment (TVLA)

One method of assessing DPA vulnerability is the Test Vector Leakage Assessment (TVLA) methodology, using Welch's t-test [9]. The t-test indicates whether two populations $Q_0$ and $Q_1$ are distinguishable. However, it cannot be used to recover a secret key, or determine the relative difficulty of a DPA attack. A confidence factor $t$ is calculated as $(\mu_0 - \mu_1)/\sqrt{(s_0^2/n_0) + (s_1^2/n_1)}$, where $\mu_0$ and $\mu_1$ are means of $Q_0$ and $Q_1$, $s_0$ and $s_1$ are standard deviations, and $n_0$ and $n_1$ are the numbers of samples. If the absolute value of $t$ is above a certain threshold (e.g., $|t| > 4.5$), we can say that the two populations are distinguishable with high confidence [21].

In this research we employ a non-specific t-test, where we select a fixed plaintext $D$, and then randomly select $D$ or random data, and feed it to the device under test (DUT). The collected traces are partitioned into two sets $Q_0$ and $Q_1$, corresponding to fixed or random data. A t-test is then performed on the two sets. If the traces are distinguishable, this indicates that the device is leaking (some sort of) information, and is possibly vulnerable to DPA.

### 3.4 FOBOS

Power analysis on cipher implementations in this research is performed using the Flexible Open-source workBench fOr Side-channel analysis (FOBOS), which includes a single "acquisition to analysis" open-source toolkit, and can use a variety of low-cost FPGA boards [5]. As shown in Fig. 2, FOBOS is composed of two FPGA boards. The control board controls the flow of test vectors to and from the host PC and device under test (DUT) board, and synchronizes trigger actions with the oscilloscope. The DUT board, in which the user instantiates a victim cipher, contains a wrapper which stores four different types of data – public data, secret data, random data, and data output – for use by an authenticated cipher, in accordance with protocol defined in [11]. In particular, the random data input (RDI) interface is used for protected ciphers, in order to perform initial share separation of public and secret data. FOBOS test vectors, including initial random data supplied to the cipher, are pre-generated in software based on valid authenticated cipher test vectors created by utilities in [6].

Our FOBOS installation uses an Agilent Technologies DSO6054A Oscilloscope, with 4 GSa/s sampling rate, and captures current fluctuations sensed by a Tektronix CT1 current probe in series with the 1.2V power supply of the DUT board, which is a Digilent Nexys-3 with Spartan-6 FPGA in this research.

FOBOS analysis software is written in Python, and enables features such as TVLA, or fine-grain leakage analysis, as described below.

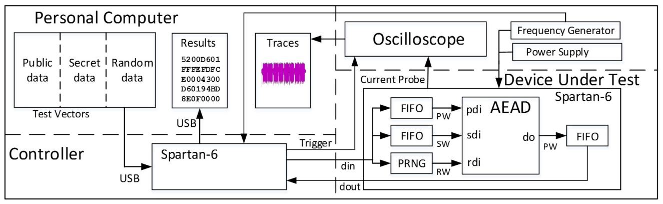

**Figure 2: FOBOS Hardware Architecture.**

In this research we extend the capabilities of FOBOS with fine-grain leakage analysis in FOBOS Profiler, which is a set of scripts that map power samples, collected in traces by the oscilloscope, to specific clock cycles and device state in the victim algorithm. This allows the user to correlate leakage, observed in a t-test, to the exact offending data or condition in the victim cipher.

The control board can be configured to trigger the oscilloscope immediately upon start of a cryptographic operation, or delayed for an arbitrary number of clock cycles. Likewise, the trigger persists for a user-specified number of clock cycles, or until the end of the operation, according to a configuration file.

The trigger provides the reference point for mapping clock cycle number to sample number in the trace. Users can generate a file that maps internal device state to clock cycles. This "state file" can be generated using a simulator tool, such as Xilinx iSim or Modelsim.

The profiler script uses the state file and power traces to display or tabulate clock transitions and the state at any clock cycle.

## 4 RESULTS

### 4.1 Assessing leakage of implementations

All t-tests in this research are conducted by FOBOS using 2,000 traces at an externally generated frequency of 500 KHz. We first conduct a t-test on an unprotected implementation of CLOC-AES. The results, shown at left in Fig. 3, indicate a t-test failure, as expected for unprotected implementations.

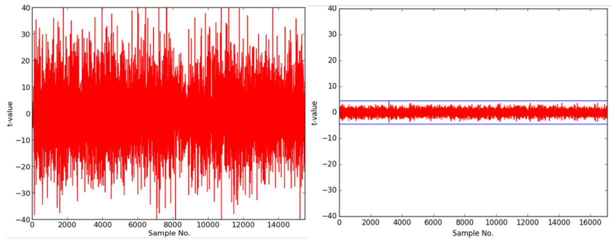

**Figure 3: Results of t-test of Unprotected CLOC-AES (left) and Unconditionally Protected CLOC-AES (right)**

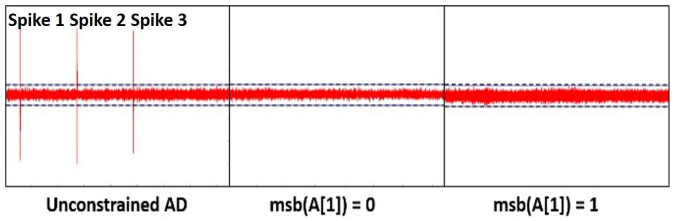

**Figure 4: Results of t-test for protected CLOC-AES, including unconstrained Associated Data (*AD*) (left), constrained to $msb(A[1]) = 0$ (center), and constrained to $msb(A[1]) = 1$ on right. T-tests at center and right are passing t-tests for the Conditionally Protected CLOC-AES.**

An examination of the CLOC specification reveals a data-dependent decision condition. As shown in Alg. 1, a tweak $h$ is performed on $S_H$ (defined in [13]) based on the most-significant-bit (msb) of the padded first word of associated data $A[1]$. Whether the algorithm is implemented in hardware or software, some decision mechanism must choose whether or not to replace $S_H[1]$ with $h(S_H[1])$. In the case of the CLOC-AES hardware implementation, this decision is made in a Finite-State Machine (FSM) controller, and communicated to a multiplexer in the cipher datapath.

**Algorithm 1** Portion of CLOC HASH algorithm with data-dependent branch, where terminology is defined in [13]

```
1: if $msb_1$(ozp(A[1])) = 1 then
2:     $S_H[1] \leftarrow h(S_H[1])$
3: end if
```

We hypothesize that the t-test will detect the data-dependent decision condition, regardless of our TI-protected implementation. To test our hypothesis, we conduct a t-test on our protected implementation of CLOC-AES with an unconstrained fixed-versus random test vector. The t-test generally passes, except for unambiguous discrete points of leakage, or "spikes," shown at left in Fig. 4.

Next, we generate two sets of alternative test vectors, where $msb(A[1])$ is constrained to be either 0 or 1, and repeat the t-tests. The results in the center and at the right of Fig. 4 show fully passing t-tests for both test vectors, which supports our hypothesis that the t-test is able to detect this data-dependent decision. Since the t-test on the conditional test vectors passes as shown in Fig. 4, we call this CLOC-AES implementation "conditionally protected".

It is worth mentioning at this point that a failure of a t-test does not prove vulnerability to DPA, nor does a passing t-test prove immunity to DPA [22]. In fact, it would be difficult to develop a power model to recover the secret key based on information leakage in the CLOC-AES hash function, since associated data is presumed to be publicly available and not encrypted. However, if one chooses to use TVLA as a standard for demonstrating increased resistance to DPA, then the only acceptable passing standard is to eliminate all sources of leakage.

### 4.2 Determining source of leakage

We next apply the FOBOS Profiler techniques as discussed above to pin-point the source of leakage. We first modify the VHDL source codes of CLOC-AES to record the controller states at each clock cycle. Specifically, we note the following instance of code in the controller in Alg. 2, and add distinguishers for "branch taken" (i.e., State `0x0C`) and "branch not taken" (i.e., State `0x0D`). We then simulate the design using Xilinx iSim, which records state activity, and the exact number of clock cycles (3,397), which consists of one authenticated decryption and three authenticated encryptions.

Graphical results generated by the profiler, reflecting analysis of spikes 1, 2, and 3 in Fig. 4, are shown in Fig. 5. Spike 1, shown at left in Fig. 5, occurs adjacent to clock cycle 236, State `0x0C`, which is a "branch taken" reflecting $msb(A[1]) = 1$. Spike 2, shown at center in Fig. 5, occurs at clock cycle 1087, State `0x0D`, which is "branch not taken" reflecting $msb(A[1]) = 0$. Finally, Spike 3, shown at right in Fig. 5, occurs at clock cycle 1928, State `0x0C`, which is "branch taken" reflecting $msb(A[1]) = 1$. Slight misalignments between samples at which the spike occurs and the actual occurrence of the branch are possibly due to FOBOS inter-trace sample misalignments (Note: Sample numbers depicted in Fig. 5 reflect a decimation rate of 10 in a compressed version of these traces). A misalignment of one or two clock cycles notwithstanding, the graphical evidence in Fig. 5 confirms the hypothesis of leakage caused by the data-dependent branch condition shown in Alg. 1.

### 4.3 Fixing the leakage

To achieve an unconditionally passing t-test, we must eliminate the data-dependent branch. This necessitates designing an $h$ tweak which always executes, regardless of $msb(A[1])$. Therefore, our goal is to define a function $g(x, y) : g(S_H[1], msb(A[1]))$, which fulfills conditions in Alg. 1 for all $x$ and $y$, but where $g(x, y)$ does not have a data-dependent branch.

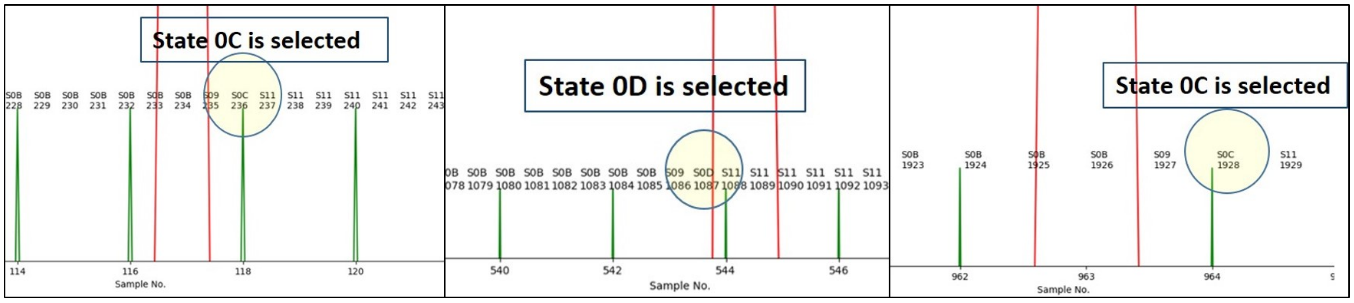

**Figure 5: Correlation of discrete leakage spikes (i.e., two tall thin red lines) to clock cycle and device state using FOBOS Profiler**

**Algorithm 2** CLOC-AES Controller Algorithm, where $msb_a$ is $msb_1(ozp(A[1]))$, $sel_h$ is a multiplexer selector, and $state\_debug$ records device state in simulation

```
1: if (msb_a = '1') then
2:     sel_h <= '1';
3:     state_debug <= x"0c";
4: else
5:     sel_h <= '0';
6:     state_debug <= x"0d";
7: end if
```

Although the $h$ tweak itself is a linear function, a branch decision, often described by a two-to-one multiplexer in register transfer level (RTL) design, in fact is a non-linear function. Therefore, we must create a hybrid 2-/3-share TI-protected non-linear transformation $g(S_H[1]_a, S_H[1]_b, msb(A[1])_a, msb(A[1])_b)$, where shares $a$ and $b$ are the 2-share TI-protected shares used in the conditionally protected implementation of CLOC-AES.

To get rid of the data-dependent branch in Alg. 1, we redefine the implicit multiplexer in Alg. 1 based on Boolean expressions using not, and, and or primitives, as $(S_H[1] \wedge A[1]) \vee (h(S_H[1]) \wedge \overline{A[1]})$. Inside the non-linear transformation itself, two shares must be expanded to three shares to meet the TI non-completeness property. This occurs inside specially-constructed 3-share TI-protected and and or modules. The resulting non-linear function with no multiplexer is shown in the right side of Fig. 6, compared to the original linear function with multiplexer, in the left of Fig. 6. Note that one bit selectors are extended to 128 bits, and that an odd number of selectors (e.g., $msb(A[1])_b$) are inverted when implementing the negation of a TI-protected Boolean expression. Additionally, the correct unmasked value of two-share output of the TI-protected or module can be recovered at any time using $result_a \oplus result_b$.

The above change is made to the conditionally protected implementation of CLOC-AES. Additionally, the code in Alg. 2 is removed from the controller. The resulting implementation is retested using FOBOS, using test vectors where $msb(A[1])$ is unconstrained. As shown at right in Fig. 3, this "unconditionally protected" implementation passes the t-test.

## 4.4 Evaluating costs of protection

The three versions of CLOC-AES, including unprotected (UnPr), conditionally protected (Pr(Cnd)), and unconditionally protected

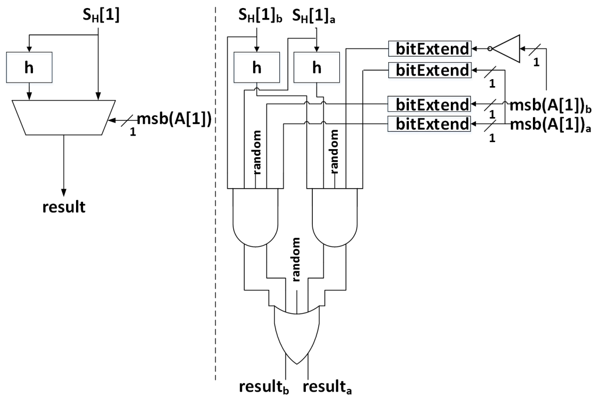

**Figure 6: Branch-dependent $h$ tweak (left) and Non-branch-dependent TI-protected non-linear transformation (right). Bus widths are 128 bits unless noted.**

(Pr(UnCnd)), are implemented in the Spartan-6 (xc6xls16csg324-3) FPGA using Xilinx 14.7 ISE. Post place-and-route results, depicted as look up tables (LUTs), combinational logic block (CLB) slices, registers, maximum frequency (MHz), throughput (TP)(Mbps), and throughput-to-area (TP/A) ratio (Mbps/LUT), are shown in Table 1. In particular, TP is defined as the output of the number of bits of ciphertext per unit time, assuming a long message consisting of many blocks of plaintext. The results show that the cost of going from unprotected to conditionally protected is a 3.0× increase in number of LUTs, a 42% reduction in TP, and an 81% reduction in TP/A ratio. This essentially represents the cost of a protected cryptographic primitive (i.e., AES), and a two-share TI-protected linear layer at the authenticated cipher layer.

The costs of achieving the unconditionally protected CLOC-AES implementation are a 3.8× increase in LUTs, 48% reduction in TP, and 86% reduction in TP/A compared to the unprotected implementation; and a 1.25× increase in LUTs, 9% reduction in TP, and 27% reduction in TP/A ratio compared to the conditionally protected implementation. Specifically, there is a 1428 LUT increase from conditional to unconditional protection; in other words, going from the t-tests of Fig. 4 to the t-test at right in Fig. 3.

Some of the additional cost of the unconditionally protected implementation is due to increased randomness required to meet

**Table 1: Comparison of CLOC-AES implementations**

| | UnPr | Pr (Cnd) | Ratio vs. UnPr | Pr (UnCnd) | Ratio vs UnPr | Ratio vs. Pr(Cnd) |
|---|---|---|---|---|---|---|
| LUT | 1915 | 5831 | 3.045 | 7259 | 3.791 | 1.245 |
| Regs | 2013 | 5063 | 2.515 | 5456 | 2.710 | 1.078 |
| Slices | 705 | 2112 | 2.996 | 2277 | 3.230 | 1.078 |
| Freq | 153.51 | 88.86 | 0.579 | 80.38 | 0.524 | 0.905 |
| TP | 95.385 | 55.214 | 0.579 | 49.945 | 0.524 | 0.905 |
| TP/A | 0.050 | 0.009 | 0.190 | 0.007 | 0.138 | 0.727 |

the TI uniformity property. On each call to the protected $h$ transformation, we provide 262 additional random bits – 256 bits for resharing from two to three shares in 128-bit TI-protected and and $\underline{\text{or}}$ modules, and 6 bits for resharing of $msb(A[1])_a$, $msb(A[1])_b$, and $\overline{msb(A[1])_b}$. Since $h$ is called at most once per authenticated encryption or decryption (i.e., hundreds of clock cycles), we accumulate this randomness in registers fed by the PRNG, and expend it all on a single clock cycle on the call to $h$.

This represents the additional cost of a large non-linear transformation at the authenticated cipher layer, which has not been previously reported in literature. Since current and future cryptographic competitions and standardization efforts evaluate potential authenticated cipher candidates on their costs of protection against SCA including DPA, we recommend that developers of cryptographic algorithms consider the significant costs of attempting to protect large non-linear transformations using algorithmic countermeasures, such as threshold implementations.

## 5 CONCLUSIONS

In this research, we demonstrated protection of an FPGA implementation of the CLOC-AES authenticated cipher against Differential Power Analysis using Threhsold Implementation (TI) countermeasures. We used Test Vector Leakage Assessment (TVLA) methodology, conducted with the Flexible Opensource workBench fOr Side-channel analysis (FOBOS), to show information leakage in an unprotected CLOC-AES implementation. We then tested a conditionally-protected implementation, and determined that DPA countermeasures are effective, with the exception of discrete leakage spikes in the t-test. Using fine-grain leakage analysis with FOBOS Profiler, we pin-pointed the source of leakage to a data-dependent branch condition in the CLOC specification. We restructured the data-dependency to a non-linear Boolean expression, and applied TI protection methodologies to the resulting non-linear transformation. A subsequent t-test confirmed that the source of leakage was eliminated, and that we achieved an unconditionally protected CLOC-AES implementation.

The costs of achieving the unconditionally protected implementation are high: 3.8× the area, 48% reduction in throughput (TP), and 86% reduction in throughput-to-area (TP/A) ratio compared to the unprotected implementation; and 1.25× the area, 9% TP reduction, and 27% TP/A ratio reduction compared to the conditionally protected implementation. In particular, the delta between the unconditionally and conditionally protected implementations is exactly equivalent to the cost of a significant non-linear transformation, located at the authenticated cipher layer and outside of the AES cryptographic primitive. The implication for future designers of authenticated ciphers is that they should carefully consider the DPA protection costs of any non-linear transformations in their specifications.

## ACKNOWLEDGMENTS

This research is based on work supported by the National Science Foundation under Grant No. 1314540, and by the Semiconductor Research Corporation under Contract No. 2017-TS-2772.

## REFERENCES

[1] Daniel J. Bernstein. 2016. Cryptographic Competitions. (Jul 2016). https://groups.google.com/forum/#!forum/crypto
[2] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. 2014. A More Efficient AES Threshold Implementation. In *Progress in Cryptology – AFRICACRYPT 2014*. 267–284.
[3] CAESAR. 2012. Competition for Authenticated Encryption: Security, Applicability, and Robustness. (2012). http://competitions.cr.yp.to/caesar.html.
[4] David Canright and Lejla Batina. 2008. A Very Compact 'Perfectly Masked' S-Box for AES. In *ACNS*.
[5] CERG. 2016. Flexible Open-source workBench fOr Side-channel analysis (FOBOS). (Oct 2016). https://cryptography.gmu.edu/fobos/
[6] CERG. 2017. Development Package for Hardware Implementations Compliant with the CAESAR Hardware API, v2.0. (Dec 2017). https://cryptography.gmu.edu/athena/index.php?id=CAESAR.
[7] William Diehl, Abubakr Abdulgadir, Farnoud Farahmand, Jens-Peter Kaps, and Kris Gaj. 2018. 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)*. 147–152.
[8] George Mason University. 2016. CERG Source Code. (Oct 2016). https://cryptography.gmu.edu/athena/index.php?id=sourcecodes
[9] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. 2011. A Testing Methodology for Side Channel Resistance Validation. NIST Non-invasive Attack Testing Workshop. (2011).
[10] Hannes Groß and Stefan Mangard. 2017. Reconciling $d+1$ Masking in Hardware and Software. *IACR Cryptology ePrint Archive* 2017 (2017), 103.
[11] Ekawat Homsirikamol, William Diehl, Ahmed Ferozpuri, Farnoud Farahmand, Panasayya Yalla, Jens-Peter Kaps, and Kris Gaj. 2016. CAESAR Hardware API. Cryptology ePrint Archive, Report 2016/626. (2016). http://eprint.iacr.org/2016/626.pdf.
[12] Ekawat Homsirikamol, Panasayya Yalla, Farnoud Farahmand, William Diehl, Ahmed Ferozpuri, Jens-Peter Kaps, and Kris Gaj. 2017. Implementer's Guide to Hardware Implementations Compliant with the CAESAR Hardware API, v2.0. (Dec 2017). https://cryptography.gmu.edu/athena/CAESAR_HW_API/CAESAR_HW_Implementers_Guide_v2.0.pdf
[13] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka, and Eita Kobayashi. 2016. CLOC and SILC. (Sep 2016). https://competitions.cr.yp.to/round3/clocsilcv3.pdf
[14] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Advances in Cryptology — CRYPTO' 99*. 388–397.
[15] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. 2011. Introduction to Differential Power Analysis. *Journal of Cryptographic Engineering* 1, 1 (01 Apr 2011), 5–27.
[16] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. 2011. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *Advances in Cryptology – EUROCRYPT 2011*. 69–88.
[17] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. 2006. Threshold Implementations Against Side-Channel Attacks and Glitches. In *Information and Communications Security*. 529–545.
[18] NIST. 2001. Federal Information Processing Standards Publication 197, Advanced Encryption Standard (AES). (2001).
[19] NIST. 2018. DRAFT Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. (Apr 2018).
[20] Niels Samwel and Joan Daemen. 2017. DPA on Hardware Implementations of Ascon and Keyak. In *Proceedings of the Computing Frontiers Conference, CF'17, Siena, Italy, May 15-17, 2017*. ACM, 415–424.
[21] Tobias Schneider and Amir Moradi. 2016. Leakage Assessment Methodology. *Journal of Cryptographic Engineering* 6, 2 (Jun 2016), 85–89.
[22] François-Xavier Standaert. 2017. 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.pdf.