TRIFLE

Designers/Submitters:

Nilanjan Datta - Indian Institute of Technology Kharagpur, India
Ashrujit Ghoshal - University of Washington, USA
Debdeep Mukhopadhyay - Indian Institute of Technology Kharagpur, India
Sikhar Patranabis - Indian Institute of Technology Kharagpur, India
Stjepan Picek - Delft University of Technology, The Netherlands
Rajat Sadhukhan - Indian Institute of Technology Kharagpur, India

nilanjan.datta@iitkgp.ac.in, ashrujit@cs.washington.edu, debdeep@cse.iitkgp.ac.in, sikhar.patranabis@iitkgp.ac.in, s.picek@tudelft.nl, rajat.sadhukhan@iitkgp.ac.in

March 29, 2019

Chapter 1

Introduction

In this document, we propose a new ThReshold Induced Fault resistant Lightweight authenticated Encryption mode, and dub it TRIFLE. TRIFLE is a nonce misuse resistant, inverse-free authenticated cipher amenable to area-efficient implementations with resistance against side-channel analysis [10] and fault injection analysis attacks [14]. This makes it a great candidate for deployment in lightweight applications with area-constrained target platforms.

It is well-documented in the cryptographic literature that side-channel analysis and fault injection analysis attacks constitute a major threat to the security of cryptographic implementations, even when the underlying cryptographic algorithm is (provably or heuristically) secure against known cryptanalytic techniques [10,14]. The threat is further amplified with the advent of pervasive computing and the Internet of Things (IoT), where a multitude of inter-connected devices in the wild that are readily accessible offer numerous attack vectors to malicious adversaries.

Countermeasures against side-channel attacks (e.g., masking/threshold implementations [3,11,12]) and fault analysis attacks (e.g., redundancies/error-correction codes [16,17]) can be highly area-consuming if implemented in ad-hoc manner. This makes it necessary to design the underlying cryptographic algorithm in a manner that allows for area-efficient side-channel and fault resilience.

In this document, we address this issue by proposing a deterministic authenticated encryption with the following desirable properties:

1. **Area-efficient Side-channel Resilience.** Our underlying block cipher is a substitution-permutation network (SPN) with specially designed CA-rule based substitution boxes (S-boxes), that can be protected using highly area-efficient threshold implementations (TI) [3,12]. In particular, TI circuits for our S-box occupy significantly lower area than those for existing lightweight block ciphers, such as PRESENT [4] and GIFT [1].

2. **Inherent Fault-attack Resilience.** We choose a specially-designed mode that makes our overall design inherently resistant against well-known fault analysis techniques. Informally, our mode makes it impossible for the adversary to execute multiple instances of the underlying block cipher using the same plaintext-key pair, which is a requirement for all known fault analysis techniques. This allows us to avoid the additional area-requirements for dedicated countermeasures such as spatial/temporal redundancies [16] and error-correction codes [17].

1.1 Notation

First, we introduce all the required notations. By $\{0,1\}^*$ we denote the set of all strings, and by $\{0,1\}^n$ the set of strings of length n. $|A|$ denotes the number of the bits in the string A. We use the notation $\oplus$ and $\cdot$ to refer the field addition and multiplication, respectively. Integer addition and multiplications are represented by $+$ and $\times$. For $A, B \in \{0,1\}^\star$, $A\|B$ to denotes the concatenation of A and B. We use the notation $V_{v-1}\|\cdots\|V_0 \overset{i}{\leftarrow} V$ to denote parsing of the string V into v vectors with $|V_j| = i$, for all $j = 1, \ldots, (v-1)$ and $|V_{v-1}| \leq i$. The expression $\mathcal{E}?\ a : b$ evaluates to a if $\mathcal{E}$ holds and b otherwise. If $m \leq n$, for $X \in \{0,1\}^n$ we denote by $\lfloor X \rfloor_m$ (resp., $\lceil X \rceil_m$) the m left-most (resp., right-most) bits of X. OZP is the function that applies optional $10^\star$ padding on n bits, i.e., $\mathsf{OZP}(X) = 0^{n-|X|-1}\|1\|X$ when $|X| < n$, and $\mathsf{OZP}(X) = X$, if $|X| = n$.

Chapter 2

Mode Specification

TRIFLE is a block cipher based authenticated encryption mode with block size $n = 128$ that receives an encryption key $K \in \{0,1\}^{128}$, a nonce $N \in \{0,1\}^{128}$, an associated data $A \in \{0,1\}^*$ and a message $M \in \{0,1\}^*$ as inputs and returns a ciphertext $C \in \{0,1\}^{|M|}$ and a tag $T \in \{0,1\}^{128}$. Here, $|M|$ denotes the length or size of M in number of bits. The corresponding verification decryption algorithms receive a key $K \in \{0,1\}^{128}$, a nonce $N \in \{0,1\}^{128}$, an associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$ and a tag $T \in \{0,1\}^{128}$ as inputs and return the plaintext $M \in \{0,1\}^{|C|}$ corresponding to C if the tag T is verified.

TRIFLE uses a block cipher E as the underlying primitive. It employs a MAC-then-Encrypt type paradigm, where CBC style authentication is done on the nonce, associated data and the plaintext to generate the tag. This tag is then used as a random IV in an output feedback mode of encryption to generate the ciphertext. Proper domain separations during the tag generation is done using constant multiplications. The formal algorithmic description of the mode is presented in Algorithm 1.

Algorithm 1 Algorithmic description of TRIFLE.

```
1: function TRIFLE.enc($N, A, M$)
2:     $T \leftarrow \mathsf{HASH}(N, A, M)$
3:     for $i = 1$ to $m$ do
4:         $T \leftarrow E_K(T)$
5:         $C_i \leftarrow T \oplus M_i$
6:     return $(C, T)$

7: function HASH($N, A, M$)
8:     $b_0 \leftarrow |A| > 0?\ 1 : 0$
9:     $b_1 \leftarrow |M| > 0?\ 1 : 0$
10:    $\mathsf{CS} \leftarrow 0^{n-2} \| b_1 \| b_0$
11:    $V \leftarrow E_K(\mathsf{CS})$
12:    $\mathsf{CS} \leftarrow \mathsf{CS} \oplus N$
13:    $T \leftarrow E_K(V \oplus N)$
14:    $(\mathsf{CS}, T) \leftarrow \mathsf{MAC}(A, T, \mathsf{CS})$
15:    $(\mathsf{CS}, T) \leftarrow \mathsf{MAC}(C, T, \mathsf{CS})$
16:    $T \leftarrow E_K(T \oplus \mathsf{CS})$
17:    return $T$
```

```
1: function TRIFLE.dec($N, A, C, T$)
2:     $V \leftarrow T$
3:     for $i = 1$ to $m$ do
4:         $M_i \leftarrow \lfloor V \rfloor_{|C_i|} \oplus C_i$
5:         $V \leftarrow E_K(V)$
6:     if $(\mathsf{HASH}(N, A, M) = T)$ then
7:         return $M$
8:     else
9:         return $\perp$

10: function MAC($D, V, \mathsf{CS}$)
11:    $D_d \| \cdots \| D_1 \leftarrow D$
12:    for $i = 1$ to $d - 1$ do
13:        $\mathsf{CS} \leftarrow \mathsf{CS} \oplus D_i$
14:        $V \leftarrow E_K(V \oplus D_i)$
15:    $\alpha \leftarrow |D_d| = n?\ 2 : 4$
16:    $V \leftarrow E_K(\alpha \cdot (V \oplus \mathsf{OZP}(D_d)))$
17:    $\mathsf{CS} \leftarrow \mathsf{CS} \oplus \mathsf{OZP}(D_d)$
18:    return $(\mathsf{CS}, V)$
```

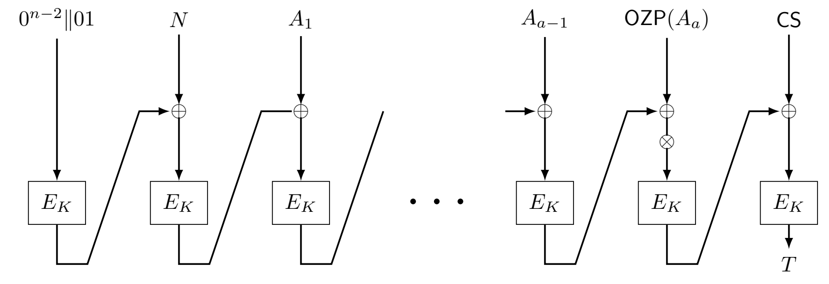

Figure 2.1: TRIFLE with a AD blocks and empty message.

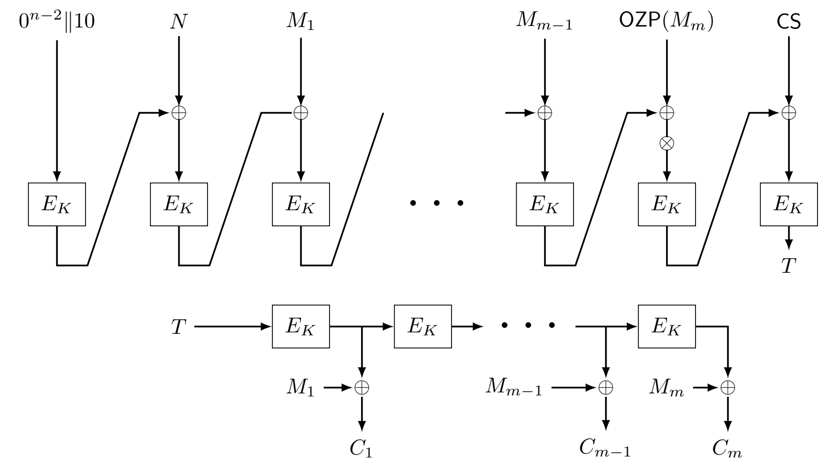

Figure 2.2: TRIFLE with empty AD and m message blocks.

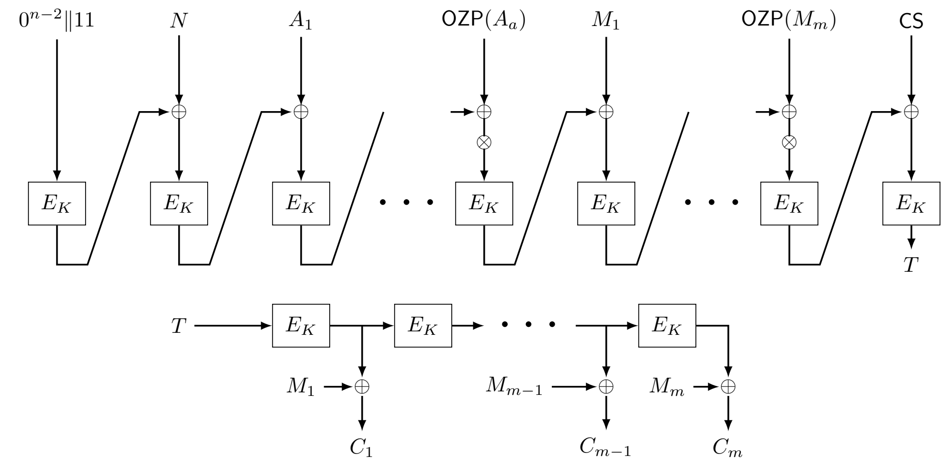

Figure 2.3: TRIFLE with a AD blocks and m message blocks.

Chapter 3

Block Cipher Specification

In this section, we propose TRIFLE-BC, a CA-based side-channel resistant 128 bit block cipher. It receives an 128 bit plaintext $X_{127}X_{126}\cdots X_0$ as the cipher state X where X_0 is the least significant bit. The cipher state can be viewed as 32 4-bit nibbles $X = W_{31}\|W_{30}\|\cdots\|W_0$. Along with the plaintext, the cipher also receives a 128-bit key $K = K_7\|K_6\|\cdots\|k_0$ as the key state, where K_i is a 16-bit word. The cipher is composed of 50 rounds and each round is composed of the following operations:

SubNibbles. TRIFLE-BC uses an invertible CA-based 4-bit S-box and apply it to every nibble of the cipher state. Description of this S-box is given in Table 3.1.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	0	C	9	7	3	5	E	4	6	B	A	2	D	1	8	F

Table 3.1: The TRIFLE-BC S-box.

Our CA-based S-box has the following properties:

- It is bijective,
- For any fixed non-zero input difference $a \in F_4^2$ and any fixed non-zero output difference $b \in F_4^2$:

$$\#\{x : F_4^2 | S(x) + S(x+a) = b\} \leq 4,$$

- For any fixed non-zero input difference $a \in F_4^2$ and any fixed non-zero output difference $b \in F_4^2$ with $\mathsf{Ham}(a) = \mathsf{Ham}(b) = 1$:

$$\#\{x : F_4^2 | S(x) + S(x+a) = b\} \leq 2,$$

- For any fixed non-zero $a \in F_4^2$ and any non-zero $b \in F_4^2$:

$$|S_b^W(a)| \leq 8,$$

- For any fixed non-zero $a \in F_4^2$ and any non-zero $b \in F_4^2$ with $\mathsf{Ham}(a) = \mathsf{Ham}(b) = 1$:

$$S_b^W(a) = \pm 4.$$

We call a 4×4 S-box satisfying the above mentioned properties as *super-optimal.*

BitPermutation. TRIFLE-BC uses an optimal bit permutation to create maximal diffusion. This bit mapping is presented in Table 3.2. Note that this permutation maps bits from bit position i of the cipher state to bit position $P(i)$, where

$$P(i) = \lfloor i/4 \rfloor + (i\%4) \times 32.$$

AddRoundKey. In this step, a 64 bit round key is extracted from the key state, and the round key is xored with $\{X_{4i+1}, X_{4i+2}\}_{i=0,\ldots,31}$ of the cipher state. The round-keys are generated using a key scheduling algorithm which updates the key state at each round using simple word-wise rotations and bit-wise rotations within a word. This key generation algorithm is similar to the one used in GIFT-128.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathsf{P}(i)$	0	32	64	96	1	33	65	97	2	34	66	98	3	35	67	99
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\mathsf{P}(i)$	4	36	68	100	5	37	69	101	6	38	70	102	7	39	71	103
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$\mathsf{P}(i)$	8	40	72	104	9	41	73	105	10	42	74	106	11	43	75	107
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$\mathsf{P}(i)$	12	44	76	108	13	45	77	109	14	46	78	110	15	47	79	111
i	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
$\mathsf{P}(i)$	16	48	80	112	17	49	81	113	18	50	82	114	19	51	83	115
i	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
$\mathsf{P}(i)$	20	52	84	116	21	53	85	117	22	54	86	118	23	55	87	119
i	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
$\mathsf{P}(i)$	24	56	88	120	25	57	89	121	26	58	90	122	27	59	91	123
i	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
$\mathsf{P}(i)$	28	60	92	124	29	61	93	125	30	62	94	126	31	63	95	127

Table 3.2: The TRIFLE-BC BitPermutation P.

AddRoundConst. In this step, a 6 bit round constant is generated using using the same 6 bit affine LFSR used in SKINNY [2]. The round constant is xored with the following 6 cipher state bits: X_{23} , X_{19} , X_{15} , X_{11} , X_7 , X_3. We also add a constant bit 1 in the most significant bit X_{127}.

Complete specification of TRIFLE-BC is presented in Algorithm 2.

Algorithm 2 TRIFLE-BC Block cipher Algorithm.

```
 1: function TRIFLE-BC$_K(X)$
 2:     $C \leftarrow 0^6$
 3:     for $i = 1$ to $50$ do
 4:         $X \leftarrow$ SubNibbles$(X)$
 5:         $X \leftarrow$ BitPermutation$(X)$
 6:         $(K, X) \leftarrow$ AddRoundKey$(K, X, i)$
 7:         $(C, X) \leftarrow$ AddRoundConst$(C, X)$
 8:     return $X$

 9: function SubNibbles$(X)$
10:     $X_{15} \| \cdots \| X_0 \overset{4}{\leftarrow} X$
11:     for $i = 0$ to $15$ do
12:         $X_i \leftarrow S(X_i)$
13:     return $X$

14: function BitPermutation$(X)$
15:     $X_{127} \| \cdots \| X_0 \overset{1}{\leftarrow} X$
16:     for $i = 0$ to $127$ do
17:         $P(i) = \lfloor i/4 \rfloor + (i\%4) \times 32$
18:         $X_{P(i)} \leftarrow X_i$
19:     return $X$
```

```
 1: function AddRoundKey$(K, X, i)$
 2:     $K_7 \| \cdots \| K_0 \overset{16}{\leftarrow} K$
 3:     $X_{127} \| \cdots \| X_0 \overset{1}{\leftarrow} X$
 4:     $U_{31} \| \cdots U_0 \leftarrow K_4 \| K_5$
 6:     $V_{31} \| \cdots \| V_0 \leftarrow K_1 \| K_0$
 7:     for $i = 0$ to $31$ do
 8:         $X_{4i+2} \leftarrow X_{4i+2} \oplus U_i$
 9:         $X_{4i+1} \leftarrow X_{4i+1} \oplus V_i$
10:     $K_7 \| \cdots \| K_0 \leftarrow K_1 \ggg 2 \| K_0 \ggg 12 \| K_7 \| \cdots \| K_2$
11:     return $(K, X)$

12: function AddRoundConstant$(C, X)$
13:     $C_5 \| \cdots \| C_0 \overset{1}{\leftarrow} C$
14:     $X_{127} \| \cdots \| X_0 \overset{1}{\leftarrow} X$
15:     $X_{127} \leftarrow X_{127} \oplus 1$
16:     $X_{23} \leftarrow X_{23} \oplus C_5$
17:     $X_{19} \leftarrow X_{19} \oplus C_4$
18:     $X_{15} \leftarrow X_{15} \oplus C_3$
19:     $X_{11} \leftarrow X_{11} \oplus C_2$
20:     $X_7 \leftarrow X_7 \oplus C_1$
21:     $X_3 \leftarrow X_3 \oplus C_0$
22:     $C_5 \| \cdots \| C_0 \leftarrow C_4 \| \cdots \| C_0 \| (C_5 \oplus C_4 \oplus 1)$
23:     return $(C, X)$
```

Chapter 4

Security

In this chapter, we present some analysis on the security of our proposal. Section 4.1 presents a brief analysis against generic attacks (assuming the underlying block cipher is a pseudo random permutation), i.e., the security of the mode. Section 4.2 presents basic cryptanalysis results of TRIFLE-BC.

4.1 Security of **TRIFLE**

Here, we describe some possible strategies to attack the TRIFLEmode, and provide a rough estimate on the amount of data and time required to mount those attacks (see Table 4.1). The data complexity of the attack quantifies the online resource requirements, and includes the total number of blocks (among all messages and associated data) processed through the underlying block cipher for a fixed master key. We would like to emphasize that the above data and time complexities remain same even if the adversary uses same nonce arbitrary number of times.

Security Model	Data complexity	Time complexity
IND-CPA	64	128
INT-CTXT	64	128

Table 4.1: Security Claims. We remark that the given values indicate the amount of data and time required to make the attack advantage close to 1.

4.1.1 IND-CPA or Privacy Security of **TRIFLE**

To attack against the privacy of TRIFLE, we assume that an adversary makes at most q encryption queries $(A^i, M^i)_{i=1..q}$ to TRIFLE with an aggregate of total σ many blocks. Let the corresponding ciphertext tag pairs be $(C^i, T^i)_{i=1..q}$. Now an adversary can distinguish our construction from a random function of same domain and range only if it observes a non-trivial collision in the inputs of the block cipher, i.e., there exists some message i and j such that any one of the following events occur:

1. $T^i = T^j$
2. $M^i_k \oplus C^i_k = T^j$
3. $M^i_k \oplus C^i_k = M^j_l \oplus C^j_l$
4. $T^i \in \{0^n, 010^{n-2}, 10^{n-1}, 110^{n-2}\}$
5. $M^i_k \oplus C^i_k \in \{0^n, 010^{n-2}, 10^{n-1}, 110^{n-2}\}$

It is easy to see that, the probability of occurrence of any one of the above mentioned events can be bounded by $O(\sigma^2/2^n)$. This is due to the fact that the HASH function we use is in fact a secure message authentication code and hence the T^i values will be random.

4.1.2 INT-CTXT or Integrity Security of TRIFLE

Given that no non-trivial collision occurs in the inputs of the block cipher, the only way that an adversary can mount a forgery is by guessing the tag which occurs with probability $q_v/2^n$. This probability along with the collision probability ensures that the probability of mounting a forgery is bounded by $O\big((\sigma^2 + q_v)/2^n\big)$.

4.2 Security of TRIFLE-BC

In this section, we analyse the security of TRIFLE-BC, showing that it resists linear and differential attacks and behaves as a pseudo random permutation under the given data and time limit.

4.2.1 Linear Cryptanalysis

Given a linear characteristic with a bias LP, the square of the correlation contribution (correlation potential) is defined as $4.(\mathsf{LP})^2$. For an adversary to mount linear cryptanalysis on a 128-bit block cipher, she would require the correlation potential to be larger than 2^{-128}. To analyse the resistance of TRIFLE-BC against linear attacks, we present the following lemma where we analyse the best linear approximation to four rounds of TRIFLE-BC:

Lemma 1 *Let LP_4 be the maximal bias of a linear approximation of four rounds of TRIFLE-BC. Then $\mathsf{LP}_4 \leq 2^{-7}$.*

Proof: According to the definition of super-optimal S-boxes, the bias of all linear approximations is less than 2^{-2} while the bias of any single-bit approximation is less than 2^{-3}. Let $\mathsf{LP}_{(4,a)}$ denote the bias of a linear approximation over 4 rounds involving a active S-boxes. Now consider the following three cases:

- $a = 4$. In this case, each round linear approximation has exactly one active S-box. The bias of each of the two S-boxes in the middle rounds is at most 2^{-3} and hence, the overall bias for a four round approximation can be bounded as follows:

$$\mathsf{LP}_{(4,4)} \ \leq \ 2^3.(2^{-2})^2.(2^{-3})^2 \ \leq \ 2^{-7}.$$

- $a = 5$. In this case, there are exactly five active S-boxes over four rounds. The optimal diffusion property of the bit-permutation ensures that three consecutive rounds cannot form the pattern 1-2-1. Consequently the number of active S-boxes is either 2-1-1-1 or 1-1-1-2, and hence

$$\mathsf{LP}_{(4,5)} \ \leq \ 2^4.(2^{-2})^4.(2^{-3})^1 \ \leq \ 2^{-7}.$$

- $a \geq 6$. If there are more than five active S-boxes,

$$\mathsf{LP}_{(4,\geq 6)} \ \leq \ 2^{a-1}(2^{-2})^a \ \leq \ 2^{-7} \text{ for any } a \geq 6.$$

This completes the proof. □

We use the above lemma directly to bound the maximal bias of a 44-round linear approximation by $2^{10}.(2^{-7}))^{11} \leq 2^{-67}$. Hence, the correlation contribution of 44-round is less than 2^{-130}. Therefore, we believe 50-round TRIFLE-BC is enough to resist against linear cryptanalysis.

4.2.2 Differential Cryptanalysis

Generally, for an adversary to mount an attack on a 128-bit block cipher using differential cryptanalysis, there must be some differential propagation with differential probability larger than 2^{-127}. The case of the differential cryptanalysis of TRIFLE-BC is handled by an MILP program that finds the best differential characteristics up to 10 rounds of TRIFLE-BC. We present the result in Table 4.2.
We use this result directly to claim that the differential probability of 50-round TRIFLE-BC is at most $(2^{-28})^5 \leq 2^{-140}$. Hence, we believe that 50 round is sufficient for our cipher to resist against differential cryptanalysis.

Round Number	4	5	6	7	8	9	10
Differential Probability	2^{-10}	2^{-13}	2^{-16}	2^{-19}	2^{-22}	2^{-25}	2^{-28}

Table 4.2: Differential Characteristic of TRIFLE-BC.

4.2.3 Integral Cryptanalysis

TRIFLE-BC S-box can be described via 10 inequalities given below:

$$\begin{aligned}
a_3 + a_2 + a_1 + a_0 - b_3 - b_2 - b_1 - b_0 &\geq 0 \\
-b_3 - b_2 + 2b_1 - b_0 + 1 &\geq 0 \\
-b_3 - b_2 - b_1 + 2b_0 + 1 &\geq 0 \\
-a_3 - a_2 - a_1 - a_0 + 3b_3 + 3b_2 + 3b_1 + 3b_0 &\geq 0 \\
-b_3 + 2b_2 - b_1 - b_0 + 1 &\geq 0 \\
2b_3 - b_2 - b_1 - b_0 + 1 &\geq 0 \\
-a_3 - a_1 - a_0 + 2b_3 + 3b_2 + 3b_1 + 3b_0 &\geq 0 \\
-a_2 - a_1 - a_0 + 3b_3 + 3b_2 + 3b_1 + 2b_0 &\geq 0 \\
-a_3 - a_2 - a_1 + 3b_3 + 3b_2 + 2b_1 + 3b_0 &\geq 0 \\
-a_3 - a_2 - a_0 + 3b_3 + 2b_2 + 3b_1 + 3b_0 &\geq 0
\end{aligned}$$

The feasible solutions of these inequalities are exactly the 54 division trails of TRIFLE-BC S-box described in Table 4.3.

Input Division Property	Output Division Property
(0,0,0,0)	(0,0,0,0)
(0,0,0,1)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(0,0,1,0)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(0,0,1,1)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(0,1,0,0)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(0,1,0,1)	(0,0,1,0),(0,1,0,0),(1,0,0,0)
(0,1,1,0)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(0,1,1,1)	(0,0,1,0),(0,1,0,0),(1,0,0,0)
(1,0,0,0)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(1,0,0,1)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(1,0,1,0)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(1,0,1,1)	(0,0,0,1),(0,0,1,0),(0,1,0,0)
(1,1,0,0)	(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)
(1,1,0,1)	(0,0,1,0),(0,1,0,0),(1,0,0,0)
(1,1,1,0)	(0,0,0,1),(0,0,1,0),(1,0,0,0)
(1,1,1,1)	(0,0,1,0),(1,1,0,0)

Table 4.3: Division Trail of TRIFLE S-box.

We have used the above mentioned division trail and found integral distinguishers up to 7 rounds.

- We have found a 4-round distinguisher for TRIFLE-BC by varying consecutive 4-bits that correspond to an S-box and fixing all other 124 bits. However, we found no distinguisher in this setting increasing the number of rounds to 5.
- We have found a 5-round distinguisher for TRIFLE-BC by varying 8-bits (two sets of consecutive 4-bits corresponding to two S-boxes) and fixing all other 120 bits. However, we found no distinguisher in this setting increasing the number of rounds to 6.

- We have found a 6-round distinguisher for TRIFLE-BC by varying 16-bits (four sets of consecutive 4-bits corresponding to two S-boxes) and fixing all other 112 bits. However, we found no distinguisher in this setting increasing the number of rounds to 7.
- We have found a 7-round distinguisher for TRIFLE-BC by varying 32-bits (eight sets of 4-bit nibbles corresponding to eight S-boxes) and fixing all other 96 bits. However, we found no distinguisher in this setting increasing the number of rounds to 8.

4.2.4 Impossible Differential Cryptanalysis

Impossible differential cryptanalysis for r rounds exploits a pair of difference Δ_1 and Δ_2 such that the state difference Δ_1 never reaches the state difference Δ_2 after r rounds.

We have implemented impossible differential search tool based on MILP, considering the differential distribution through the S-box. We have exhaustively tested input and output differences satisfying the following conditions.

- The input difference activates exactly one of the 32 S-boxes.
- The output difference activates exactly one of the 32 S-boxes.

Overall, we have For the first condition, there are $32 \times 15 = 480$ many input differences as well as 480 many output differences. Hence, we have tested $480 \times 480 = 2,30,400$ pairs of input and output differences and the search results show that $1,15,200$ pairs are actually impossible for 4 rounds. We extend the search for 5 rounds and have not found any impossible differentials.

Chapter 5

Design Rationale

5.1 Choice of the Mode

TRIFLE is nonce-based authenticated encryption, where the same nonce can be used for all the queries without degrading any security. It can be viewed as a deterministic authenticated encryption where the nonce is considered as the first block of the associated data, and in this respect the mode provides maximal robustness against the misuse of nonce. It is an inverse-free authenticated encryption mode. Both encryption and decryption of the algorithm do not require any decryption call to the underlying block cipher TRIFLE-BC. This ensures a significant reduction of the overall hardware footprint, especially in the combined encryption-decryption implementations.

A notable feature of our chosen mode is that it makes the overall design inherently resistant against well-known fault analysis techniques, such as Differential Fault Analysis (DFA) [14] and Differential Fault Intensity Analysis (DFIA) [7]. We provide a high level argument to justify this claim.

We begin by pointing out each of the aforementioned fault analysis techniques requires two or more invocations of a block cipher encryption algorithm on the *same plaintext-key pair*, with the fault being injected in one or more of such invocations, such that the adversary learns the differential between the correct and faulty outputs.

Now, observe that as per our choice of mode (see Figures 2.3, 2.2 and 2.1), the only way for the adversary to ensure such invocations is to inject a fault in the i^{th} encryption block in the circuit that outputs the tag T, and to try and adjust the input message into the $(i+1)^{\text{th}}$ so that the eventual tag T remains unchanged. The differential between the original and final inputs to the $(i+1)^{\text{th}}$ block would thus leak the output fault differential to the adversary, and would allow him to launch a DFA/DFIA-based attack.

However, this attack fails in our setting, since every time the adversary changes one or more message blocks, the corresponding checksum block CS also gets altered, resulting in a faulty tag $T' \neq T$. In other words, the adversary fails to recover the desired differential between the correct and faulty ciphertexts. Thus we use the mode to make the overall design inherently resistant against fault attacks, and avoid the need for additional countermeasures such as spatial/temporal redundancies [16] and error-correction codes [17].

5.2 Choice of the Block cipher

5.2.1 Choice of the S-box

We intend to use a 4×4 S-box that can be easily protected against side-channel attacks using compact area-efficient implementations. More specifically, we focus on low area *threshold implementations* (TI) [3]. TI is a state-of-the-art and widely used masking technique proposed by Nikova et al. [12] as a countermeasure against Differential Power Attacks (DPA) [10]. What sets TI apart from most masking techniques is the security it guarantees even in non-ideal circuits where glitches have shown to result in leakage in more conventional masking schemes [11]. TI works under extremely relaxed assumptions on the underlying leakage which are more achievable in practical scenarios.

In our design, we propose using a 4×4 S-boxes derived using cellular automata (CA) rules. Our proposal is based on the fact that CA-based S-boxes have recently been shown to allow threshold implementations with very low area footprint [8]. In fact, as we demonstrate in Table 5.1, the CA-based S-box chosen for our design can be protected using a TI circuit with lower area footprint than the S-boxes of the PRESENT and GIFT block ciphers (all figures reported correspond to post-place and route results on a Virtex5 (xc5vlx50) FPGA. We have synthesized TI implementation of our S-box design along with PRESENT and GIFT S-boxes on the same Virtex5 platform and compared the result as shown in Table 5.1.

S-box	Slice Registers	Slice LUTs
PRESENT	13	30
GIFT	13	24
TRIFLE	9	17

Table 5.1: Area-comparison for TI of S-boxes: Post-place and route results on a Virtex5 (xc5vlx50) FPGA

We provide a high level intuition for why CA-based S-boxes allow area-efficient TI. A cellular automaton is (informally) a finite state machine whose state transitions are based on simple local rules. Prior studies have extensively analyzed the scope of realizing complex functions via repeated iterations of such simple rules [18]. A recent work by Picek et al. [13] explores the possibility of designing cryptographically optimal 4×4 S-boxes from simple 4×1 CA-based rules. The idea is to iterate over a single instance of the CA rule, while cyclically shifting the input bits, to obtain one output bit of an S-box at a time.

Now, when designing a TI circuit for the overall S-box, we first design a TI circuit for the core CA rule by decomposing the input and output bits into as few shares as possible, and then iterate over this core unit by cyclically permuting over the input bits. Note that in other non-CA-based 4×4 S-boxes, such as those of PRESENT and GIFT, one would need to implement a separate TI-circuit for each of the 4 output bits. This obviously incurs more area overhead as compared to CA-based S-boxes, where a single TI circuit suffices for all four output bits.

Additionally, if the core CA-rule is a high degree polynomial in the input bits, we follow a technique proposed in [15] to further decompose it into two or more low-degree polynomials. We then design TI circuits for these "decomposed polynomials", and cascade them in series to maintain functional equivalence. The decomposition-based strategy is motivated by the fact that cascading multiple low degree polynomials typically gives rise to TI implementations with lower area footprint than a single high-degree polynomial [8, 15].

Note that while saving on area requirements, we do not compromise on the cryptographic security of our S-box. In particular, our S-box belongs to a class of S-boxes that we refer to as *super-optimal.* The notion of super-optimal S-boxes is already explained in Section 3. The reason of considering the super-optimal S-box is to reduce the number of rounds of a block-cipher that uses the 4-bit S-box. Note that, all the CA-based S-boxes have branch number 2 and do not exhibit BOGI property. Hence, we need some additional good properties of the S-boxes, which is obtained via this notion. Our first goal is to identify CA based Super-optimal S-boxes. We have found 192 CA based S-boxes which are Super-optimal. As we have seen in section 4, this property essentially ensures that around 50 rounds of the underlying block cipher would have very good linear and differential characteristics. With an optimal (but not super-optimal) S-box, we would require at least 64 rounds to have the desired security. We choose the particular S-box as it is also optimal from all considered S-boxes with respect to the confusion coefficient [6] and modified transparency order [5], which are two properties increasing the side-channel resilience of a cipher.

5.2.2 Choice of the Bit-Permutation

In a substitution-permutation network of 128 bits (with 4 bit S-boxes), a bit-permutation achieves optimal diffusion if all the following properties hold:

- the input of an S-box in round r comprises output bits from 4 different S-boxes in round $r-1$,
- the output of an S-box in round r is distributed across the inputs of 4 different S-boxes in round $r+1$,
- the output of the S-box group $\{4i, 4i+1, 4i+2, 4i+3\}$ in round r entirely constitutes the input for the S-box group $\{i, i+8, i+16, i+24\}$ in round $r+1$.

Currently, we just need any bit-permutation with optimal diffusion. Consequently, a bit-permutation such as the one used in GIFT should be sufficient. However, as we are not exploiting the BOGI property here, we can construct a new bit-permutation similar to one used for PRESENT using the underlying group mapping as presented in Table 7 of [1].

5.2.3 Choice of the Key Schedule and Add-round-subkey

We use the same key scheduling, AddRoundKey and AddRoundConst as used in GIFT. The main rationale behind this key scheduling and addition of the round keys are given below:

1. The key scheduling is done using simple word-wise rotations and bit-wise rotations within a word. Thus the key schedule can be viewed as a simple wire shuffle and requires no additional hardware area.
2. The round keys are xored into only half of the cipher state. This saves a significant amount of hardware area in a round-based implementation.
3. The round key xors are done uniformly at the same bit positions of each nibble. This makes it software friendly and bitslice implementation more efficient.
4. The round constants are generated using a 6-bit affine LFSR that requires only a single XNOR gate per update. Each of the 6 bits is xored to a different nibble to break the symmetry.

Chapter 6

Threshold Implementation in Hardware

In this chapter, we present an overview of the TI circuit for the TRIFLE-BC S-box. In particular, we first illustrate how the CA rule describing the S-box can be expressed as a composition of two CA rules of lower algebraic degree. We then describe the individual TI circuits corresponding to these rules. Finally, we present the overall hardware architecture, where the TI circuits corresponding to these rules are cascaded in series to preserve the functionality while ensuring side-channel resistance.

6.1 CA Rule Decomposition

We define class (a, b, c) of CA-rule as a tuple of three elements, where a, b and c denotes number of degree 3, degree 2 and degree 1 terms respectively in the CA-rule. The CA-rule that describes our S-box has algebraic degree 3 and belongs to class $(3, 2, 2)$. We express a degree 3 function into a combination of degree 2 functions in order to construct *composite TI* targeting low area footprint and power consumption. We then identify uniform and non-complete sharing for each of these sub-functions and finally cascade them to obtain the final output. We decompose the function from degree 3 to degree 2 as shown below:

$$\boldsymbol{f(x_0, x_1, x_2, x_3) = x_0 * x_1 * x_2 \oplus x_0 * x_2 * x_3 \oplus x_1 * x_2 * x_3 \oplus x_0 * x_2 \oplus x_2 * x_3 \oplus x_0 \oplus x_1}$$

$$\begin{aligned} f = f_3(x_2, f_1, f_2) &= x_2 * f_1 \oplus f_2 \\ f_1(x_0, x_1, x_3) &= x_0 * x_1 \oplus x_0 * x_3 \oplus x_1 * x_3 \oplus x_0 \oplus 1 \\ f_2(x_0, x_1, x_2, x_3) &= x_2 * x_3 \oplus x_0 \oplus x_1 \oplus x_2 \end{aligned}$$

6.2 TI Decomposition

In this section, we illustrate the uniform three share decomposition of every function obtained in Section 6.1. Let s_i denotes the i^{th} share. Then we follow the below nomenclature to denote the shares:

$$x_0 = x_0^1 + x_0^2 + x_0^3, \quad x_1 = x_1^1 + x_1^2 + x_1^3, \quad x_2 = x_2^1 + x_2^2 + x_2^3, \quad x_3 = x_3^1 + x_3^2 + x_3^3,$$

$$f_1 = f_1^1 + f_1^2 + f_1^3, \quad f_2 = f_2^1 + f_2^2 + f_2^3, \quad f_3 = f_3^1 + f_3^2 + f_3^3$$

Therefore, the decomposition of f_1, f_2 and f_3 are

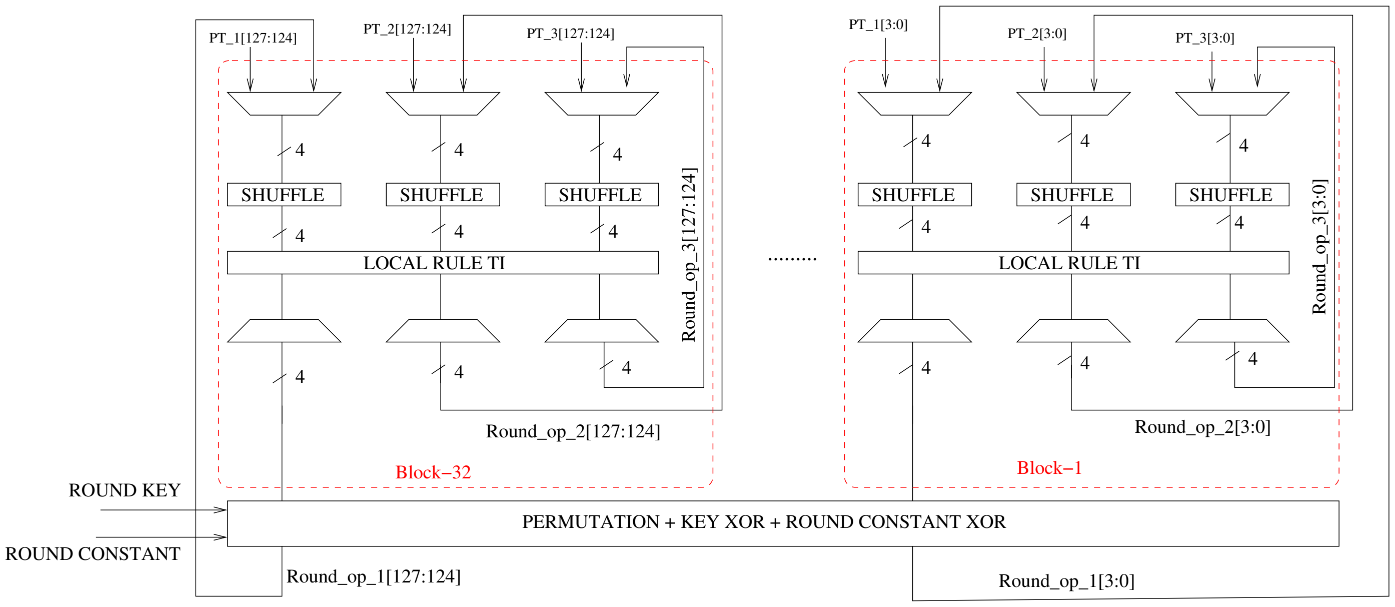

Figure 6.1: TRIFLE Architecture

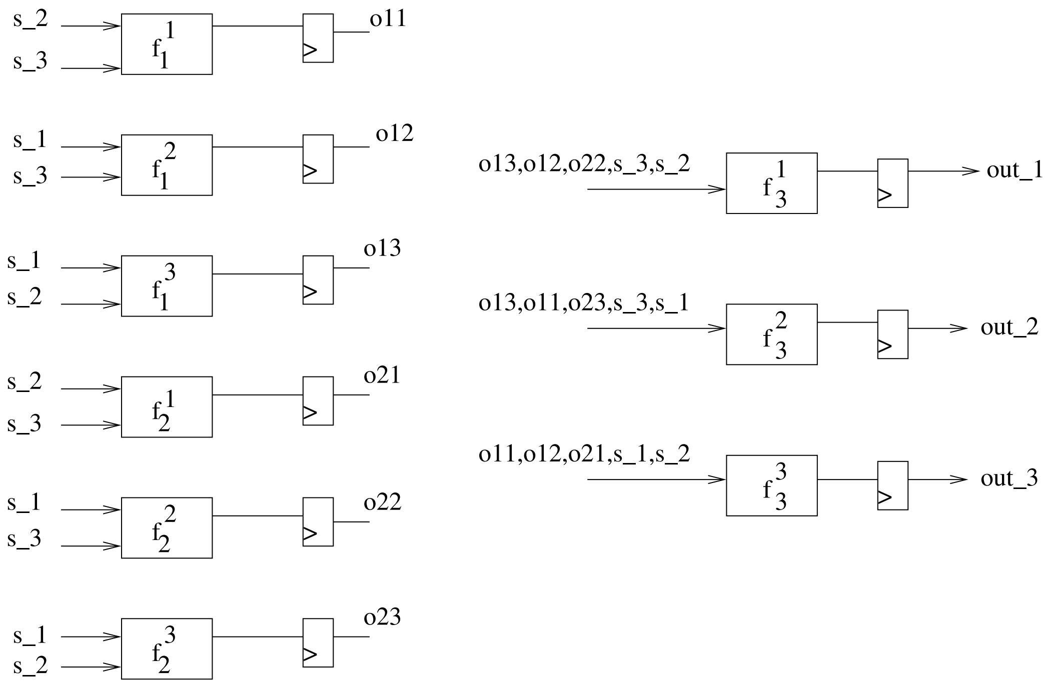

Figure 6.2: Local Rule TI of TRIFLE

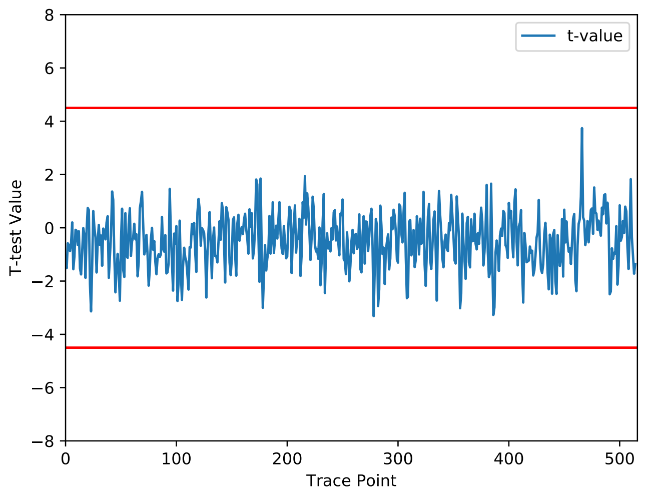

Figure 6.3: TVLA on 3-TI Implementation on TRIFLE

$$
\boxed{\begin{aligned}
f_1^1 &= 1 \oplus x_0^3 \oplus x_0^2 * x_1^3 \oplus x_0^2 * x_1^2 \oplus x_0^3 * x_1^2 \oplus x_0^2 * x_3^3 \oplus x_0^2 * x_3^2 \oplus x_0^3 * x_3^2 \oplus x_1^2 * x_3^3 \oplus x_1^2 * x_3^2 \oplus x_1^3 * x_3^2 \\
f_1^2 &= x_0^1 \oplus x_0^3 * x_1^1 \oplus x_0^3 * x_1^3 \oplus x_0^1 * x_1^3 \oplus x_0^3 * x_3^1 \oplus x_0^3 * x_3^3 \oplus x_0^1 * x_3^3 \oplus x_1^3 * x_3^1 \oplus x_1^3 * x_3^3 \oplus x_1^1 * x_3^3 \\
f_1^3 &= x_0^2 \oplus x_0^1 * x_1^2 \oplus x_0^1 * x_1^1 \oplus x_0^2 * x_1^1 \oplus x_0^1 * x_3^2 \oplus x_0^1 * x_3^1 \oplus x_0^2 * x_3^1 \oplus x_1^1 * x_3^2 \oplus x_1^1 * x_3^1 \oplus x_1^2 * x_3^1 \\
f_2^1 &= x_2^2 * x_3^2 \oplus x_2^2 * x_3^3 \oplus x_2^3 * x_3^2 \oplus x_0^2 \oplus x_1^2 \oplus x_2^2 \\
f_2^2 &= x_2^3 * x_3^3 \oplus x_2^1 * x_3^3 \oplus x_2^3 * x_3^1 \oplus x_0^3 \oplus x_1^3 \oplus x_2^3 \\
f_2^3 &= x_2^1 * x_3^1 \oplus x_2^1 * x_3^2 \oplus x_2^2 * x_3^1 \oplus x_0^1 \oplus x_1^1 \oplus x_2^1 \\
f_3^1 &= x_2^2 * f_1^2 \oplus x_2^2 * f_1^3 \oplus x_2^3 * f_1^2 \oplus f_2^2 \\
f_3^2 &= x_2^3 * f_1^3 \oplus x_2^3 * f_1^1 \oplus x_2^1 * f_1^3 \oplus f_2^3 \\
f_3^3 &= x_2^1 * f_1^1 \oplus x_2^1 * f_1^2 \oplus x_2^2 * f_1^1 \oplus f_2^1
\end{aligned}}
$$

6.3 Hardware Architecture and Implementation Details

The hardware architecture of a round implementation of TRIFLE is shown in Figure 6.1. It uses 32 blocks where each block processes PT_i and $Round\_op\_i$ denotes i-th share of plaintext and round output respectively, where $1 \leq i \leq 3$. The TI of local rule for TRIFLE is shown in Figure 6.2 which takes s_i as i-th input share and produces out_i as i-th output share. We have synthesized TI of our datapath including key schedule on Virtex-5 (xc5vlx50) platform using ISE Design suite (v14.2) and observed that TRIFLE consumes 1489 slice-registers and 1877 slice-LUTs.

6.4 Test Vector Leakage Assessment Methodology(TVLA) Analysis

To evaluate the security of our design we have tested it on SAKURA-G platform with Spartan-6 LX75 logic. We have performed *non-specific* Welch t-test [9] on our design by collecting 100000 traces.We have used Tektronix MSO54-C011756 oscilloscope collecting sampling frequency of $1.2Gs/s$ and the design running at $45.4Mhz$. We collected power traces for one round cipher operation having around 520 sample points and the t-test plot is shown in Figure 6.3. It is evident from the Figure 6.3 that the t-test(t) value is within the prescribed range of $-4.5 \leq t \leq +4.5$.

Bibliography

[1] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 321–345, 2017.

[2] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 123–153, 2016.

[3] Begül Bilgin. Threshold implementations: as countermeasure against higher-order differential power analysis. 2015.

[4] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *CHES 2007*, pages 450–466, 2007.

[5] Kaushik Chakraborty, Sumanta Sarkar, Subhamoy Maitra, Bodhisatwa Mazumdar, Debdeep Mukhopadhyay, and Emmanuel Prouff. Redefining the transparency order. *Designs, Codes and Cryptography*, 82(1):95–115, Jan 2017.

[6] Yunsi Fei, Qiasi Luo, and A. Adam Ding. A statistical model for dpa with novel algorithmic confusion analysis. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, pages 233–250, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[7] Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa M. I. Taha, and Patrick Schaumont. Differential fault intensity analysis. In *FDTC 2014*, pages 49–58, 2014.

[8] Ashrujit Ghoshal, Rajat Sadhukhan, Sikhar Patranabis, Nilanjan Datta, Stjepan Picek, and Debdeep Mukhopadhyay. Lightweight and side-channel secure 4 4 s-boxes from cellular automata rules. *IACR Trans. Symmetric Cryptol.*, 2018(3):311–334, 2018.

[9] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. P.: A testing methodology for side-channel resistance validation, niat, 2011.

[10] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.

[11] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked aes hardware implementations. In *CHES 2005*, pages 157–171. Springer, 2005.

[12] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *ICICS 2006*, pages 529–545. Springer, 2006.

[13] Stjepan Picek, Luca Mariot, Bohan Yang, Domagoj Jakobovic, and Nele Mentens. Design of s-boxes defined with cellular automata rules. In *CF 2017*, pages 409–414, 2017.

[14] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In *CHES 2003*, pages 77–88, 2003.

[15] Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-channel resistant crypto for less than 2, 300 GE. *J. Cryptology*, 24(2):322–345, 2011.

[16] Akashi Satoh, Takeshi Sugawara, Naofumi Homma, and Takafumi Aoki. High-performance concurrent error detection scheme for AES hardware. In *CHES 2008*, pages 100–112, 2008.

[17] Tobias Schneider, Amir Moradi, and Tim Güneysu. Parti - towards combined hardware countermeasures against side-channel and fault-injection attacks. In *CRYPTO 2016*, pages 302–332, 2016.

[18] Stephen Wolfram. Cellular automata as models of complexity. *Nature*, 311(5985):419, 1984.