Misusing Misuse-Resistance in APE

Dhiman Saha[1], Sukhendu Kuila[2], Dipanwita Roy Chowdhury[1]

[1]Department of Computer Science and Engineering, IIT Kharagpur, India
{dhimans,drc}@cse.iitkgp.ernet.in
[2]Department of Mathematics, Vidyasagar University, India
babu.sukhendu@gmail.com

Abstract. Misuse-resistance has increasingly become an important property of authenticated ciphers. It is particularly useful in lightweight cryptographic applications where maintaining a nonce generator entails high overhead. The ongoing CAESAR competition also encourages this feature and addresses it in detail in the selection portfolio. In this work we showcase how misuse-resistance can be *misused* in the context of differential fault analysis of on-line authenticated encryption schemes like APE which is a submission to CAESAR and a member of the PRIMATEs family of authenticated ciphers. Using the misuse, we finally present a diagonal fault attack[1] on APE-80 that is able to reduce the key-search space from 2^{160} to 2^{25} using just two random diagonal faults. Increasing the number of faults to 4 results in the unique identification of the key with a high probability.

1 Introduction

The idea of nonce-based encryption was formalized by Rogaway in [14]. The primary condition to be fulfilled is the uniqueness of the nonce in every instantiation of the cipher. All security claims rely on this premise. An interesting consequence of a unique nonce is the automatic protection from Differential Fault Analysis (DFA). Fault analysis constitutes one of the most popular forms of side channel attacks. The prospect of using faults to attack cryptographic hardware was first explored by Boneh et. al. [8, 9] in 1996. Biham and Shamir introduced Differential fault analysis (DFA) in [6] which was later successfully applied on the Advanced Encryption Standard (AES). One of the primary assumptions of DFA is the ability to induce faults in the intermediate state of the cipher while *replaying the encryption with the same plaintext*. Herein, comes the role of the unique nonce which prohibits the replaying criterion. Nonces have been used in Public-Key cryptography to thwart fault attacks. The famous Bellcore attack [9, 12] on RSA-CRT signatures can be prevented if the message is padded with a random nonce which is recoverable only when verifying a correct signature. Though Coron et. al. have shown that in [10] that in some limited setting these nonces can be tackled, the techniques used rely on theoretical constructs which may not directly work in their private-key counterparts. Thus, nonce-based encryption as professed by Rogaway seem to be have an in-built protection against DFA. In practice, however, ensuring the uniqueness of the nonce incurs high overhead specifically in lightweight cryptographic applications where resources may be highly constrained. Hence, nonce reuse or misuse becomes an important issue that should be addressed while designing crypto primitives like authenticated encryption (AE). On the other hand this implies that DFA may again become relevant. In this work we explore this prospect by showcasing a technique that reinstates the replaying criterion of DFA on the authenticated cipher APE.

In FSE 2014, Andreeva et. al. introduced a scheme named Authenticated Permutation-based Encryption (APE) [3] targeted for lightweight cryptography. It is the first permutation-based AE scheme that is resistant against nonce misuse. APE is basically a mode of operation which iterates

[1] The fault attack presented in this work has been communicated to a conference with proceedings.

a fixed permutation in a manner that is inspired from the Sponge [5] construction. In [3], the authors instantiated APE with permutations of lightweight hash functions like Spongent, Quark and Photon. However, in 2014, the authors reintroduced APE along with GIBBON and HANUMAN as part of the authenticated encryption family PRIMATEs [1] which is one of the submissions to the ongoing CAESAR competition. Unlike [3], in [1] the authors proposed an indigenous permutation called PRIMATE to serve as the underlying permutation for PRIMATEs. In terms of the internal state size the permutation has two variants : PRIMATE-80 and PRIMATE-120. The PRIMATE permutation family is inspired from FIDES [7] authenticated cipher and structurally follows the round function of the Rijndael block cipher [11].

Finally, in this work, we present ESCAPE - a differential fault analysis of APE. It is interesting to note that, APE drops two important aspects of FIDES, firstly, the assumption of a nonce-respecting adversary and secondly, the final truncation operation. Our research reveals that both these changes result in efficient differential fault attacks. We first show how misuse resistance can be exploited to *repeat* the encryption on the same plaintext. Thus dropping the nonce constraint opens up the scheme to fault attacks, which require an attacker to be able to observe faulty ciphertexts by injecting faults while *repeating* the same encryption. We next mount diagonal fault attack[15] which has been stated in literature [13] as one of the most effective DFA on AES. We exploit the fact that PRIMATE closely follows the structure of AES round function. However, the inclusion of last round MixColumns transformation makes direct application of diagonal attack as described in [15] inefficient. In order to overcome this we come up with a tweak using the linearity of the MixColumns and ShiftRows operations. Finally, removal of the final truncation of FIDES in APE helps in improving the attack further by using the knowledge of the last block of the ciphertext. The results presented here are with reference to PRIMATE-80. However, all the claims can be easily extended to PRIMATE-120. The contribution of this work is summarized below:

- Show how misuse resistance can be misused to make an AE scheme vulnerable to differential fault analysis.
- Present the first fault analysis that exploits a Sponge based mode of operation.
- Efficient adaptation of the classical diagonal fault attack on APE
- Reduce average key-space from 2^{160} to 2^{25} and 1 for two and four faults respectively.

The rest of the paper is organized as follows: The notations used in this work along with a brief description of APE is given in section 2. The idea of misusing misuse-resistance in the light of DFA and fault diffusion in the internal state of APE permutation PRIMATE are discussed in section 3. Section 4 introduces the proposed diagonal attack - ESCAPE. The concluding remarks are furnished in section 5.

2 Preliminaries

2.1 The Design of PRIMATE

PRIMATE has two variants in terms of size : PRIMATE-80 (200-bit permutation) and PRIMATE-120 (280-bit) which operate on states of (5×8) and (7×8) 5-bit elements respectively. The family consists of four permutations p_1, p_2, p_3, p_4 which differ in the round constants used and the number of rounds. All notations introduced in this section are with reference to PRIMATEs-80 with the APE mode of operation.

Definition 1 *Let* $\mathbb{T} = \mathbb{F}[x]/(x^5 + x^2 + 1)$ *be the field* $\mathbb{F}_{2^5}$ *used in the PRIMATE MixColumn operation. Then a* ***word*** *is defined as an element of* $\mathbb{T}$.

Definition 2 *Let $\mathbb{S} = (\mathbb{T}^5)^8$ be the set of (5×8)-word matrices. Then the internal* ***state*** *of the PRIMATE-80 permutation family is defined as an element of $\mathbb{S}$. We denote a state $s \in \mathbb{S}$ with elements $s_{i,j}$ as $[s_{i,j}]_{5,8}$.*

$$s = [s_{i,j}]_{5,8}, \text{ where} \begin{cases} s_{i,j} \in \mathbb{T} \\ 0 \leq i \leq 4, \ 0 \leq j \leq 7 \end{cases} \tag{1}$$

In the rest of the paper, for simplicity, we omit the dimensions in $[s_{i,j}]_{5,8}$ and use $[s_{i,j}]$ as the default notation for the 5×8 state. We denote a column of $[s_{i,j}]$ as $s_{*,j}$ while a row is referred to as $s_{i,*}$. We now describe in brief the design of PRIMATE permutation. APE instantiates p_1 which is a compositions of 12 round functions.

$$\begin{aligned} p_1 : \mathbb{S} \longrightarrow \mathbb{S}, \qquad & p_1 = \mathcal{R}_{12} \circ \mathcal{R}_{11} \circ \cdots \circ \mathcal{R}_1 \\ \mathcal{R}_r : \mathbb{S} \longrightarrow \mathbb{S}, \qquad & \mathcal{R}_r = \alpha_r \circ \mu_r \circ \rho_r \circ \beta_r \end{aligned}$$

where $\mathcal{R}_r$ is a composition of four bijective functions on $\mathbb{S}$. The index r denotes the r^{th} round and may be dropped if the context is obvious. Here, the component function β represents the non-linear transformation SubBytes which constitutes word-wise substitution of the state according to predefined S-box.

$$\beta_r : \mathbb{S} \longrightarrow \mathbb{S}, \qquad s = [s_{i,j}] \longmapsto [S(s_{i,j})]$$

where $S : \mathbb{T} \longrightarrow \mathbb{T}$ is the S-box given in Table 1. The transformation ρ corresponds to ShiftRows which cyclically shifts each row of the state based on a set of offsets.

$$\rho_r : \mathbb{S} \longrightarrow \mathbb{S}, \qquad s = [s_{i,j}] \longmapsto [s_{i,(j-\sigma(i)) \bmod 8}]$$

where, $\sigma = \{0, 1, 2, 4, 7\}$ is the ShiftRow offset vector and $\sigma(i)$ defines shift-offset for the i^{th} row. The MixColumn operation, denoted by μ, operates on the state column-wise. μ is actually a left-multiplication by a 5×5 matrix (M_μ) over the finite field $\mathbb{T}$.

$$\begin{aligned} \mu_r : \mathbb{S} &\longrightarrow \mathbb{S} \\ s = [s_{i,j}] &\longmapsto s' = [s'_{i,j}] \\ s'_{*,j} &= M_\mu \times s_{*,j} \end{aligned}$$

The last operation of the round function is α which corresponds to the round constant addition. The constants are the output $\{C_1, C_2, \cdots, C_{12}\}$ of a 5-bit LFSR and are xored to the word $s_{1,1}$ of the state $[s_{i,j}]$. The APE mode of operation is depicted in Fig. 1.

$$\begin{aligned} \alpha_r : \mathbb{S} &\longrightarrow \mathbb{S} \\ [s_{i,j}] &\longmapsto [s'_{i,j}] \\ s'_{i,j} &= \begin{cases} s_{i,j} \oplus C_r \text{ if } i, j = 1 \\ s_{i,j}, \text{ Otherwise} \end{cases} \end{aligned}$$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(x)$	1	0	25	26	17	29	21	27	20	5	4	23	14	18	2	28
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$S(x)$	15	8	6	3	13	7	24	16	30	9	31	10	22	12	11	19

Table 1: The PRIMATE 5-bit S-box

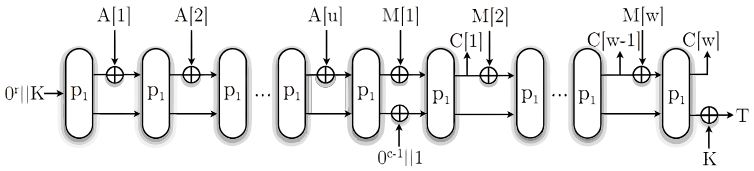

Fig. 1: The APE mode of operation [1](encryption)

2.2 Notations

Definition 3 *A **diagonal** of a state ($s = [s_{i,j}]$) is the set of words which map to the same column under the Shift-Row operation.*

$$d_k = \{s_{i,j} : \rho(s_{i,j}) \in s_{*,k}\},\ \textit{where} \begin{cases} k = (j - \sigma(i)) \bmod 8 \\ \sigma = \{0, 1, 2, 4, 7\} \end{cases} \tag{2}$$

Definition 4 *A **differential state** is defined as the element-wise XOR between a state $[s_{i,j}]$ and the corresponding faulty state $[s'_{i,j}]$.*

$$s'_{i,j} = s_{i,j} \oplus \delta_{i,j},\ \forall\ i, j \tag{3}$$

δ fully captures the initial fault as well as the dispersion of the fault in the state. In this work we assume induction of random faults in some diagonal of a state. Thus, if the initial fault occurs in diagonal $d_k \in s$, the differential state is of the following form :

$$\delta_{i,j} = \begin{cases} f : f \xleftarrow{R} \mathbb{T} \setminus 0, \text{ if } k = j - \sigma(i) \text{ for at least one } (i, j) \\ 0, \text{ Otherwise} \end{cases} \tag{4}$$

If $\exists j : \delta_{i,j} = 0\ \forall i$ then $\delta_{*,j}$ is called a *pure* column, otherwise $\delta_{*,j}$ is referred to as a *faulty* column.

Definition 5 *A **Hyper-state** of a state $s = [s_{i,j}]$, denoted by $s^h = [s^h_{i,j}]$, is a two-dimensional matrix, where each element $s^h_{i,j}$ is a non-empty subset of $\mathbb{T}$, such that s is an element-wise member of s^h.*

$$s^h = \begin{bmatrix} s^h_{00} & s^h_{01} & \cdots & s^h_{07} \\ s^h_{10} & s^h_{11} & \cdots & s^h_{17} \\ \vdots & \vdots & \ddots & \vdots \\ s^h_{40} & s^h_{41} & \cdots & s^h_{47} \end{bmatrix} \quad \textit{where} \begin{cases} s^h_{i,j} \subset \mathbb{T},\ \ s^h_{i,j} \neq \varnothing \\ s_{i,j} \in s^h_{i,j}\ \forall i, j \end{cases} \tag{5}$$

The significance of a hyper-state s^h is that the state s is in a way 'hidden' inside it. This means that if we create all possible states taking one word from each element of s^h, then one of them will be exactly equal to s. We now define a transformation ρ' on a hyper-state s^h which is analogous to ρ defined in the PRIMATE round functions.

Definition 6 *The **Hyper-state ShiftRow** transformation (ρ') corresponds to cyclically shifting each row of s^h based on the predefined set of offsets σ.*

$$\rho' : \bigtimes_{j=0}^{7} s^h_{i,j} \longrightarrow \bigtimes_{j=0}^{7} s^h_{i,(j-\sigma(i)) \bmod 8}\ \forall i$$

$$s^h_{i,*} \longmapsto s^h_{i,(*-\sigma(i)) \bmod 8}\ \forall i$$

It is interesting to note that, every word in the state $\rho(s)$ will be a member of the corresponding element of $\rho'(s^h)$, thereby implying that $\rho'(s^h) = (\rho(s))^h$.

Definition 7 *The size of a hyper-state s^h, denoted by $\left|s^h\right|$ is the maximum number of the states that can be constructed by selecting a single word from each element of the hyper-state such that every state formed is an element-wise member of s^h.*

$$\left|s^h\right| = \prod_{i,j=0}^{4,7} |s^h_{i,j}| \tag{6}$$

The notion of **Hyper-state** will be used while we execute the INBOUND phase of ESCAPE described in subsection 4.1.

Definition 8 *If s^h be a hyper-state of s, then **Kernel** of a column $s^h_{*,j} \in s^h$, denoted by $\mathcal{K}^{s^h_{*,j}}$, is defined as the cross-product of $s^h_{0,j}, s^h_{1,j}, \cdots, s^h_{4,j}$.*

$$\mathcal{K}^{s^h_{*,j}} = \left\{ w_k : w_k^T \in \bigtimes_{i=0}^{4} s^h_{i,j},\ 1 \leq k \leq \prod_{i=0}^{4} \left|s^h_{i,j}\right| \right\}$$

Subsequently, Kernel of the entire hyper-state is the set of the Kernels of all of its columns: $\mathcal{K}^{s^h} = \{\mathcal{K}^{s^h_{*,0}}, \mathcal{K}^{s^h_{*,1}}, \cdots, \mathcal{K}^{s^h_{*,7}}\}$

Here, w_k^T represents the transpose of w_k, thereby implying that w_k is a column vector. One should note that $s_{*,j} \in \mathcal{K}^{s^h_{*,j}}\ \forall j$. Thus each column of s is contained in each element of $\mathcal{K}^{s^h}$. We now define an operation μ' over the Kernel of a hyper-state which is equivalent to μ that operates on a state.

Definition 9 *The**Kernel-MixColumn** transformation (μ') is the left multiplication of M_μ to each element of each $\mathcal{K}^{s^h_{*,j}} \in \mathcal{K}^{s^h}$.*

$$\mu'(\mathcal{K}^{s^h_{*,j}}) = \{M_\mu \times w_i,\ \forall w_i \in \mathcal{K}^{s^h_{*,j}}\}$$
$$\mu'(\mathcal{K}^{s^h}) = \{\mu'(\mathcal{K}^{s^h_{*,0}}), \mu'(\mathcal{K}^{s^h_{*,1}}), \cdots, \mu'(\mathcal{K}^{s^h_{*,7}})\}$$

An important implication is that $\mu'(\mathcal{K}^{s^h}) = \mathcal{K}^{\mu(s)^h}$. The notion of **Kernel** will be used while we execute the OUTBOUND phase of ESCAPE described in subsection 4.2.

3 Misusing Misuse-Resistance

The fault model assumed in this work states that: The attacker can induce random uni-word faults in the internal state of APE permutation p_1. The primary aim here is to produce faulty collisions of the tag. A faulty collision is not a real collision. In a faulty collision, the attacker induces a fault in the state of the cipher so that two different plaintexts produce the same tag. We later show that the ability to collide the tag will result in highly efficient fault attacks on the overall scheme. It is interesting to note that the attacker is able to produce faulty collisions by exploiting two very desirable properties of authenticated ciphers : firstly, the misuse-resistance and secondly, the online nature. In the next paragraph we describe the process of producing faulty collisions in the AE scheme APE.

The designers of APE in their revised[2] submission document [2] have emphasized that APE is misuse-resistant up to a common prefix. As per [4] this means that for an AE scheme, E :

[2] It is interesting to mention that in FSE 2014 paper [3] and in original submission [2] of PRIMATEs the authors mentioned that 'the use of nonce is optional'. In this scenario, the ability to produce faulty collisions would hold automatically.

$K \times \left(\{0,1\}^n\right)^+ \rightarrow \left(\{0,1\}^n\right)^+$ if nonce and associated data are *same* for two plain-texts then the outputs are same for a common prefix. So, the first $|P|$ bits are same for $E_k(P||X)$ and $E_k(P||X')$ for any $P, X, X' \in (\{0,1\}^n)^+$. From, the point of view of the attacker this implies that the plaintexts must vary at least in one block. The attacker now arbitrarily chooses the first message $P||X = (x_0||x_1||\cdots||x_i||\cdots||x_w), x_i \in \{0,1\}^n \ \forall i$ and adaptively chooses the next $P||X' = (x_0||x_1||\cdots||x'_i||\cdots||x_w)$ in such a way that $\exists i \ x'_i \neq x_i$. The attacker manipulates x'_i using the fact that APE is an online-cipher and using the assumed fault model. Let $E_k(P||X) = (y_0||y_1||\cdots||y_i||\cdots||y_w), y_i \in \{0,1\}^n \ \forall i$. Now, while processing the second message $(P||X')$, the attacker induces a fault at the output of APE in y_{i-1} and gets y'_{i-1}. He now prepares the i^{th} block of the second message as $x'_i = y_{i-1} \oplus y'_{i-1} \oplus x_i$. This means that the input to the permutation is $x'_i \oplus y'_{i-1} = x_i \oplus y_{i-1}$. This ensures that $E_k(P||X) = E_k(P||X')$ for all $x_i \in X$ except the block where the fault is induced. This also implies that $TAG(P||X) = TAG(P||X')$ thereby producing a faulty collision. The process is depicted in Fig. 2. It is worth mentioning that by virtue of the above event, the inputs to the internal permutation PRIMATE while processing both X and X' remain same. Thus one can infer that the ability to produce faulty collision is equivalent to the ability to replay the encryption with the same nonce, associated data and the same plain-text. We later show that this makes APE vulnerable to efficient differential fault analysis attacks. In the rest of the paper, we assume that replaying encryption with the same plain-text is possible and refrain from explicitly mentioning it again unless it is absolutely necessary. The next section is a direct consequence of the faulty collisions produced earlier.

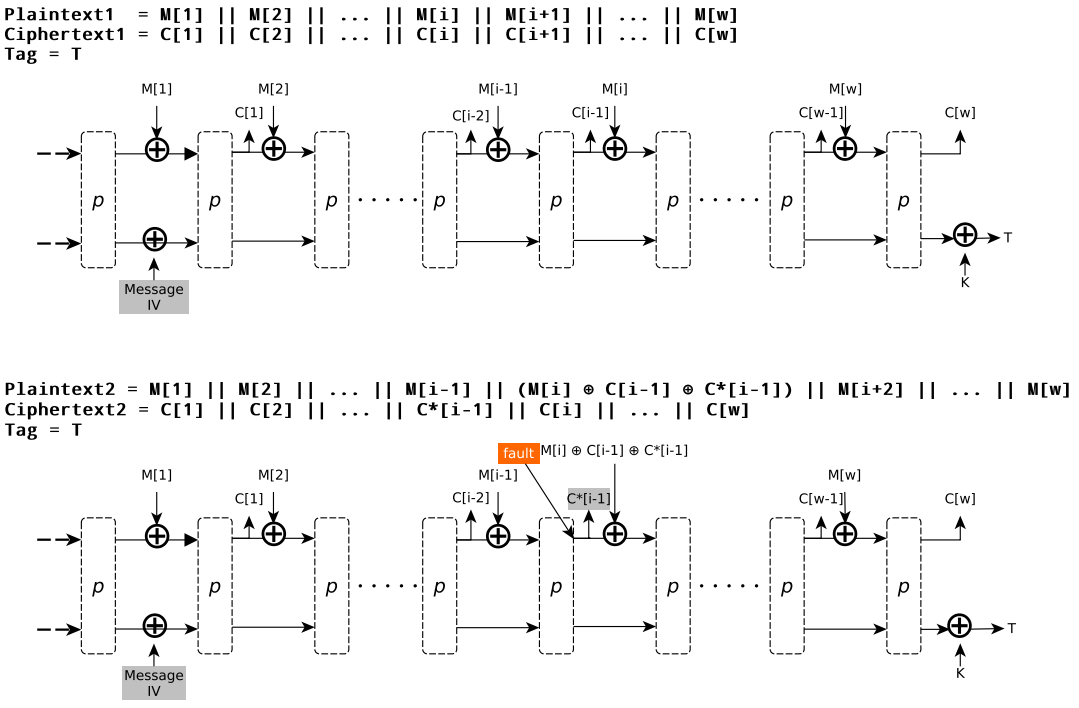

Fig. 2: Exploiting Misuse-resistance to get faulty collisions in APE[3]

[3] According to APE specification, the 'Message IV' differs depending on the nature of the length of the message. This attack is independent of the actual value of the 'Message IV'.

3.1 Fault diffusion in PRIMATE permutation

In this section we describe the induction and diffusion of faults in the last iteration of APE. In the last section, we demonstrated the ability to replay the encryption with the same plain-text. We now build upon this and induce a secondary fault while replaying the encryption. In this way we are able to produce a faulty tag that is related to the original tag. In fact, our intention is to study the fault diffusion in the differential state of the PRIMATE permutation. Next we exploit the differential state to mount efficient diagonal fault attacks on APE. The secondary fault induction and subsequent differential state formation are illustrated in Fig 3. One can see that the fault is induced in the input of Round 10 of the PRIMATE permutation during the last iteration before tag generation. The logic behind this will be clear from the following important property of fault diffusion in the internal state of PRIMATE.

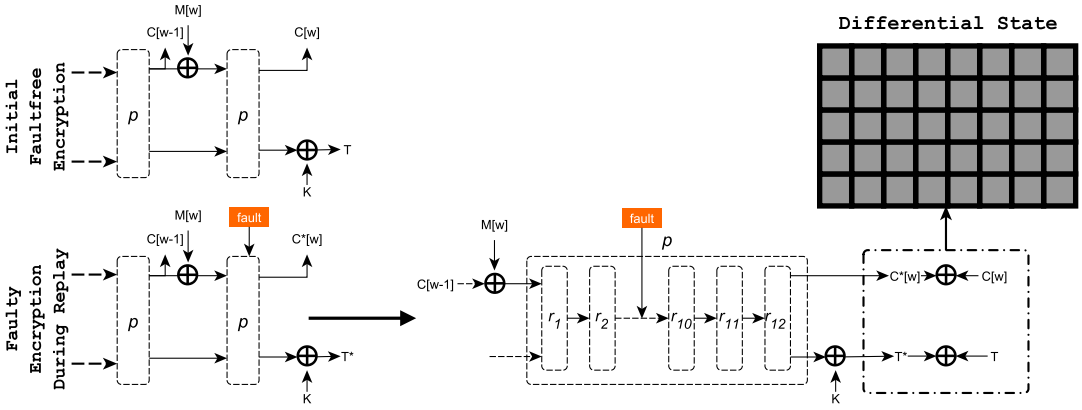

Fig. 3: Fault induction in APE and the differential state

Property 1 *If a single diagonal* (d_k) *is faulty at the start of* $\mathcal{R}_{r-2}$ *then there are exactly three pure columns after* β_r.

Analysis: This property is attributed to the non-square nature of the state matrix. To observe this we need to first look at how the diagonal fault diffuses in the state in the $(r-2)^{th}$ round. Let us denote the differential state at the input of $\mathcal{R}_{r-2}$ as $s = [s_{i,j}]$. Here, we are not concerned about the actual value of s, rather we track how the fault structurally disperses in it. At the beginning of $\mathcal{R}_{r-2}$ only diagonal d_k is faulty. In the following analysis we have omitted the transformation α, since round-constant addition has no effect on the differential state.

- Fault diffusion in $\mathcal{R}_{r-2}$
  - β_{r-2} : No diffusion, fault limited to same diagonal.
  - ρ_{r-2} : Fault shifts from diagonal d_k to column $s_{*,k}$.
  - μ_{r-2} : Intra column diffusion. Fault diffuses within $s_{*,k}$.

$$d_k \xrightarrow{\beta_{r-2}} d_k \xrightarrow{\mu_{r-2} \circ \rho_{r-2}} s_{*,k} \tag{7}$$

- Fault diffusion in $\mathcal{R}_{r-1}$
  - β_{r-1} : No diffusion, fault limited to $s_{*,k}$.

- ρ_{r-1} : Fault shifts from $s_{*,k}$ to five words $\{s_{i,(k-\sigma(i)) \bmod 8} : 0 \leq i < |\sigma|\}$.
- μ_{r-1} : Fault spreads to each column $s_{*,k-\sigma(i)}$.

$$c_k \xrightarrow{\beta_{r-1}} c_k \xrightarrow{\rho_{r-1}} \{s_{i,(k-\sigma(i)) \bmod 8}\} \xrightarrow{\mu_{r-1}} \{s_{*,(k-\sigma(i)) \bmod 8}\} \tag{8}$$

– Fault diffusion in $\mathcal{R}_r$
- β_r : No diffusion, fault limited to same columns as after μ_{r-1}.

$$\{s_{*,(k-\sigma(i)) \bmod 8}\} \xrightarrow{\beta_r} \{s_{*,(k-\sigma(i)) \bmod 8}\} \tag{9}$$

From (7), (8) and (9) we have the following relation between the faulty diagonal d_k at the start $\mathcal{R}_{r-2}$ and the faulty columns after β_r.

$$d_k \xrightarrow{\beta_r \circ \mathcal{R}_{r-1} \circ \mathcal{R}_{r-2}} \{s_{*,(k-\sigma(i)) \bmod 8} : 0 \leq i < |\sigma|\} \tag{10}$$

For PRIMATE-80, $\sigma = \{0, 1, 2, 4, 7\}$, implying that $|\sigma| = 5$. From (10), we have $|\{s_{*,(k-\sigma(i)) \bmod 8}\}| = |\sigma| = 5$. Thus a single faulty diagonal before $\mathcal{R}_{r-2}$ results in five faulty columns and respectively $8 - 5 = 3$ pure columns at the end of β_r. An example of the fault diffusion with the initial fault in diagonal d_1 is given is depicted in Fig. 4. ■

In the next section we introduce ESCAPE, an adaptation of the classical diagonal fault attack [15] on AES.

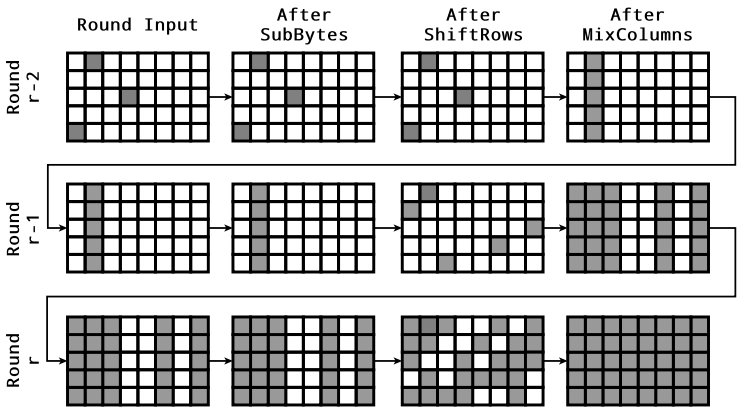

Fig. 4: 3-round fault diffusion with source fault at diagonal d_1

4 ESCAPE : An efficient Diagonal attack on APE

The ESCAPE attack proceeds in two phases namely, the INBOUND and OUTBOUND phase. Both phases result in large-scale reduction in target key-search space. We first describe the INBOUND phase.

4.1 The INBOUND Phase

The first task in the INBOUND phase is to invert the differential output of APE up to β_{12}. While doing so we exploit the following property of APE.

Property 2 *If an attacker knows the correct and the faulty outputs*[4] *from the last iteration of APE, he can use the difference between them to find the differential state after β_r, r being the last round of PRIMATE permutation.*

Analysis: This property holds because the transformations α, μ and ρ are linear and so is their composition. Let us denote the linear map as $L_r = \alpha_r \circ \mu_r \circ \rho_r$. L_r is bijective and hence invertible. Let the correct and the faulty outputs of the last iteration of APE be x and x' respectively. Also let the outputs of the $(r-1)^{th}$ round in the respective cases be y and y'. We now have the following derivation:

$$\begin{aligned} L_r^{-1}(x \oplus x') &= L_r^{-1}((x \oplus k) \oplus (x' \oplus k)) \\ &= L_r^{-1}(x \oplus k) \oplus L_r^{-1}(x \oplus k) \\ &= \beta_r(y) \oplus \beta_r(y')[\because \beta_r \circ L_r(y) = x \oplus k] \end{aligned}$$

The above derivation shows that knowledge of the output differential state leads an attacker to the differential state after the last round SubBytes operation. However, as β is non-linear, the attacker can no longer deterministically penetrate further inside the last round. ■

In case of p_1, $r = 12$. So using property (2) the attacker reaches the differential state at the end of β_{12}. From this knowledge he tries to guess the source of the fault i.e., the faulty diagonal at the start of $\mathcal{R}_{10}$. For this he exploits a very important property of PRIMATE which surfaces because of the non-square nature of the state matrix.

Property 3 *There exists a bijection between the position of faulty diagonal before $\mathcal{R}_{r-2}$ and the position of pure columns of the state at the end of β_r*

Analysis: This is evident from (10). Since $\{(k - \sigma(i)) \bmod 8, 0 \leq i < |\sigma|\}$ is unique for each k, so the position of a faulty diagonal will correspond to a unique set of five faulty (respectively three pure) columns and vice-versa. This property is important for ESCAPE, since it aids in detecting the source of the fault just by observing the differential state at the output of APE. As there are a total of 8 diagonals, the diagonal detection reduces the attacking complexity by a factor of 8 and also implies the location independence of the induced fault. ■

In case of APE this implies that based on the diagonal which is faulty at the start of $\mathcal{R}_{10}$, one can predict the word inter-relations at the end of $\mathcal{R}_{11}$. Moreover, the diagonal principle as mentioned in [15] states that *multi-word faults which are confined to one diagonal before $\mathcal{R}_{10}$ are equivalent and result in the same word inter-relations at the end of $\mathcal{R}_{11}$*. For example, if the faulty diagonal is d_0, then the corresponding relation matrix is given in Fig. 5. The relation matrix shows how words of the differential state are related at the end of $\mathcal{R}_{11}$. The empty columns in the relation matrix represent the columns which have been unaffected by the fault injected before $\mathcal{R}_{10}$. As shown in Fig. 5, by virtue of the diagonal principle all the faults in d_0 are equivalent in terms of the resulting word inter-relations. The relation matrices for other diagonals are depicted in Fig 6. We now exploit these relations to reduce the state search-space.

Let the differential state after β_{12} computed using property (2) be $\delta = [\delta_{i,j}]$ and the corresponding correct and faulty states be s and s' respectively. Let the relation matrix be denoted by

[4] This refers to the entire state where the rate part forms the *last ciphertext block* and the capacity part forms the *tag* (Refer Fig. 1).

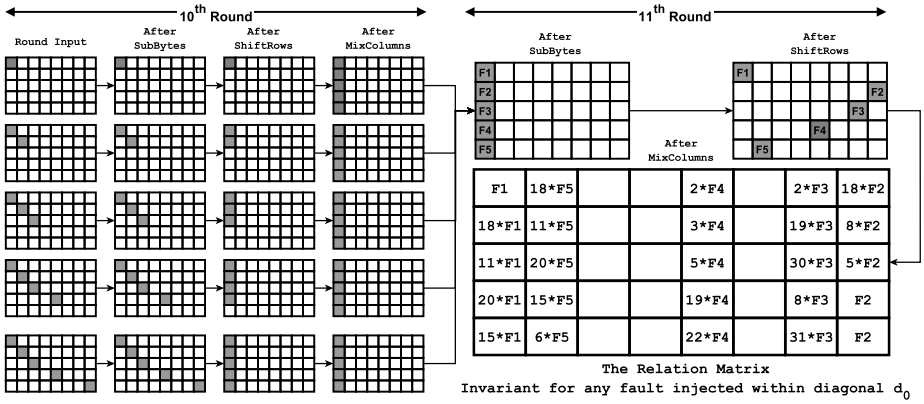

Fig. 5: Equivalence of different kinds of faults confined to diagonal d_0 at the input of 10^{th} round of p_1

$\eta = [\eta_{i,j}]$. The attacker now guesses the actual values of s and uses δ to get the values of s'. He can guess any two words $(s_{i,j}, s_{k,j})$ from the same column $s_{*,j}$ and use the corresponding entries $(\eta_{i,j}, \eta_{k,j})$ from the relation matrix to form an equation of the form given in (11).

$$\eta_{i,j}^{-1} \times (\beta^{-1}(s_{i,j}) \oplus \beta^{-1}(s_{i,j} \oplus \delta_{i,j})) = \eta_{k,j}^{-1} \times (\beta^{-1}(s_{k,j}) \oplus \beta^{-1}(s_{k,j} \oplus \delta_{k,j})) \tag{11}$$

For the case of diagonal d_0, if the attacker chooses $(s_{1,0}, s_{2,0})$, he gets equation (12). It must be recalled that all multiplications and inverse operations are carried out in the finite field $\mathbb{T}$. Using these equations, the attacker verifies if the guessed words of s are correct. The guessed words for each $s_{i,j}$ that satisfy the equations form the candidate-vector for that word. All these candidate-vectors constitute the hyper-state, s^h (Definition 5) of the correct state s. The pseudo-code for the hyper-state generation in given below:

$$18^{-1} \times (\beta^{-1}(s_{1,0}) \oplus \beta^{-1}(s_{1,0} \oplus \delta_{1,0})) = 11^{-1} \times (\beta^{-1}(s_{2,0}) \oplus \beta^{-1}(s_{2,0} \oplus \delta_{2,0})) \tag{12}$$

procedure GenHyperState(δ, η) ▷ $\delta \rightarrow$ Differential State after β_{12}
  Denote correct state after β_{12} as $s = [s_{i,j}]$.
  for all (i, j) **do** ▷ Initialize hyper-state
    $s^h_{i,j} = \{0, 1, \cdots, 31\}$
  end for
  for all $\delta_{*,j} \in \delta : \eta_{*,j} \neq \varnothing$ **do** ▷ Process only faulty columns
    for all $(\delta_{i,j}, \delta_{k,j}) \in \delta_{*,j}$ **do**
      for all $(a, b) \in s^h_{i,j} \times s^h_{k,j}$ **do**
        Set $(s_{i,j}, s_{k,j}) = (a, b)$
        if $\eta_{i,j}^{-1} \times (\beta^{-1}(s_{i,j}) \oplus \beta^{-1}(s_{i,j} \oplus \delta_{i,j})) \neq$ ▷ Verify Equation (11)
            $\eta_{k,j}^{-1} \times (\beta^{-1}(s_{k,j}) \oplus \beta^{-1}(s_{k,j} \oplus \delta_{k,j}))$ **then**
          $s^h_{i,j} = s^h_{i,j} - \{a\}$
          $s^h_{k,j} = s^h_{k,j} - \{b\}$
        end if
      end for
    end for

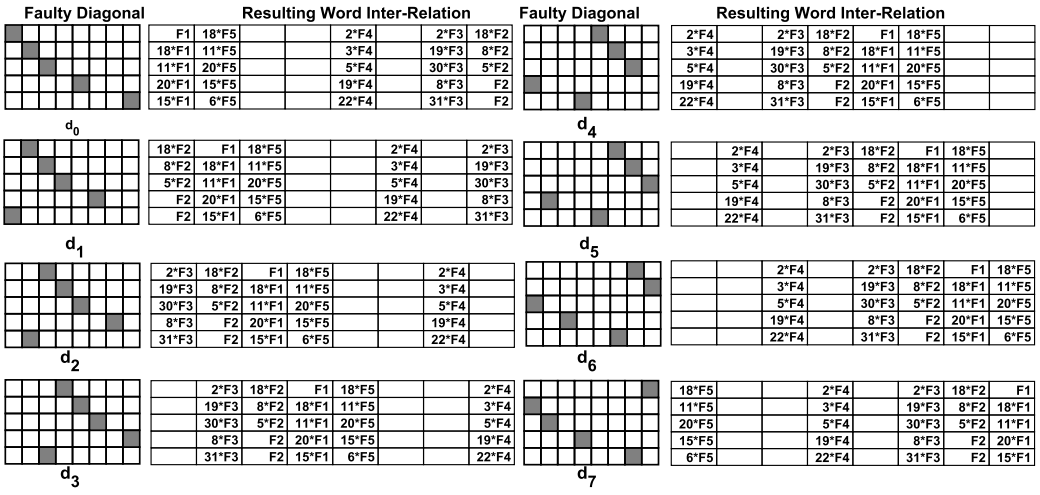

Fig. 6: Word inter-relations at the start of $\mathcal{R}_r$ due to corresponding faulty diagonal at the start of $\mathcal{R}_{r-2}$ (Empty relations refer to the pure columns)

17: **end for**
18: **return** s^h
19: **end procedure**

The formation of the hyper-state completes the INBOUND phase. It must be noted that the attacker has no clue about the pure columns of the state as he cannot get any relation to verify them. So for the pure columns the attacker search space is exhaustive. This gives the following lower bound on the size of the state-space.

Lemma 1. *At the end of the* INBOUND *phase, the size (Definition 7) of the hyper-state s^h is at least 2^{75}.*

$$\left|s^h\right| \geq 2^{75}$$

Proof. This follows from property (1) which states that there will be exactly three pure-columns at the end of $\mathcal{R}_{11}$. For each $s^h_{i,j} \in s^h$ such that $s^h_{*,j}$ is a pure column $|s^h_{i,j}| = 2^5$. Each pure-column contributes five such vectors to s^h and there are three such columns. Thus we have 15 such $s^h_{i,j}$'s each contributing a factor of 2^5 to the size of the hyper-state.

$$\begin{aligned}
\left|s^h\right| &= \prod_{i,j=0}^{4,7} |s^h_{i,j}| \\
&= \prod_{\substack{i=0 \\ \forall j \in \mathcal{P}}}^{4} |s^h_{i,j}| \times \prod_{\substack{i=0 \\ \forall j \notin \mathcal{P}}}^{4} |s^h_{i,j}| \\
&\geq \prod_{\substack{i=0 \\ \forall j \in \mathcal{P}}}^{4} |s^h_{i,j}| = (2^5)^{15} = 2^{75}
\end{aligned}$$

where $\mathcal{P}$ is the set of all pure-columns at the end of $\mathcal{R}_{11}$. ■

In the next subsection, we show that during the OUTBOUND phase the attacker can further reduce $\left|s^h\right|$ by reducing each $|s^h_{i,j}|, j \in \mathcal{P}$, thereby reducing the search space for the correct state (correspondingly, for the key). The following algorithm gives an overview of the INBOUND phase.

1: **procedure** INBOUND(δ) ▷ $\delta \rightarrow$ Differential State
2: $\delta' = \rho_{12}^{-1}(\mu_{12}^{-1}(\delta))$ ▷ Applying Property (2)
3: $d_f \xleftarrow{\text{Property (3)}} \delta'$ ▷ Trace back faulty diagonal
4: $\eta = [\eta_{i,j}] \xleftarrow{\text{Load relation matrix}} d_f$. ▷ Refer Fig. 6
5: $s^h \leftarrow$ GENHYPERSTATE(δ', η)
6: **return** s^h
7: **end procedure**

4.2 The OUTBOUND Phase

This phase exploits the mode of operation of APE. Particularly, it exploits the knowledge of the last cipher-text block that forms the rate-part of the internal state of PRIMATE. The following property plays a pivotal role in this phase.

Property 4 *If the state before α_{12} be $t = [t_{i,j}]$, then the attacker knows the actual value of $t_{0,*}$*

Analysis: This property is attributed to the APE mode of operation. According to APE, after processing the last message block the rate part (first row) of the internal state after $\mathcal{R}_{12}$ of PRIMATE is output in the clear as the last cipher-text block while the capacity part is xored with the key to form the tag. Also one can recall that in α the round constant is xored with word $(1, 1)$ only while the rest of the state is unchanged. Now, let us denote the last output of APE by $a = [a_{i,j}]$ also let $k = [k_{i,j}]$ denote the key-state. Then we have:

$$a_{i,j} = \begin{cases} t_{i,j} & \text{if } (i = 0) \implies (k_{0,*} = \mathbf{0}) \\ a_{i,j} \oplus C_{12} \oplus k_{i,j}, & \text{if } (i, j) = (1, 1) \\ a_{i,j} \oplus k_{i,j}, & \text{otherwise} \end{cases} \tag{13}$$

where c_{12} is the round constant for $\mathcal{R}_{12}$ in p_1. Since α_{12} has no effect on $t_{0,*}$ and since by Equation (13) $a_{0,*} = t_{0,*}$, so the last cipher-text block gives the attacker the actual value of $t_{0,*}$. This property plays a central role in the OUTBOUND phase and its use results in significant reduction of the key search space. ■

The steps of the OUTBOUND phase are enlisted below. It can be recalled that the INBOUND phase gives us the hyper-state (s^h) of the state s after the SubBytes operation in $\mathcal{R}_{12}$.

1. The attacker starts the OUTBOUND phase by applying Hyper-state ShiftRow transformation (Definition 6) on the hyper-state obtained from the INBOUND phase.
$$s^h \xrightarrow{\rho'} (\rho(s))^h$$
2. The next step is to compute the Kernel for $(\rho(s))^h$: $\mathcal{K}^{(\rho(s))^h}$. The concept of Kernel was introduced in Definition 8.
$$(\rho(s))^h \xrightarrow{\text{Compute Kernel}} \mathcal{K}^{(\rho(s))^h}$$
3. Then the attacker applies the Kernel-MixColumn transformation on the Kernel computed in the last step.
$$\mathcal{K}^{(\rho(s))^h} \xrightarrow{\mu'} \mathcal{K}^{(\mu(\rho(s)))^h}$$
4. Next comes the reduction step. In this step the attacker applies property (4). It can be noted that $\mathcal{K}^{\mu(\rho(s))^h}$ represents the kernel for the hyper-state of $\mu(\rho(s))$. i.e., the state just before the application of α_{12}. Now let $t = \mu(\rho(s))$. Then by property (4) the actual value of $t_{0,*}$ is known. This knowledge is used to reduce the size of each $\mathcal{K}^{t^h}_{*,j} \in \mathcal{K}^{t^h}$. The following pseudocode illustrates the reduction procedure.

1: **procedure** ReduceKernel($\mathcal{K}^{t^h}, t$)
2: **for** $j = 0 : 7$ **do**
3: **for all** $\{e_0, e_1, e_2, e_3, e_4\}^T \in \mathcal{K}^{t^h}_{*,j}$ **do**
4: **if** $e_0 \neq t_{0,j}$ **then**
5: $\mathcal{K}^{t^h}_{*,j} = \mathcal{K}^{t^h}_{*,j} - \{e_0, e_1, e_2, e_3, e_4\}^T$
6: **end if**
7: **end for**
8: **end for**
9: $\mathcal{K}^{t^h}_{red} = \mathcal{K}^{t^h}$
10: **return** $\mathcal{K}^{t^h}_{red}$
11: **end procedure**

The cross-product of $\mathcal{K}^{t^h}_{red}$ gives the final reduced state-space for t. The final key-space is given by the following expression:

$$\mathbb{K} = \left\{ k : k = a \oplus \alpha_{12}(w),\ \forall w \in \bigtimes_{j=0}^{7} \mathcal{K}^{t^h}_{*,j}_{red} \right\}$$

The following algorithm summarizes the Outbound phase. In the next sub-section we outline the complete attack.

1: **procedure** Outbound(s^h, t)
2: $(\rho(s))^h \leftarrow \rho'(s^h)$
3: $\mathcal{K}^{(\rho(s))^h} \leftarrow (\rho(s))^h$
4: $\mathcal{K}^{t^h} \leftarrow \mu'(\mathcal{K}^{\rho(s)^h})$
5: $\mathcal{K}^{t^h}_{red} \leftarrow$ ReduceKernel($\mathcal{K}^{t^h}, t$)
6: **return** $\mathcal{K}^{t^h}_{red}$
7: **end procedure**

4.3 The Complete Attack

The Escape attack in its complete form takes into account multiple faults and further reduces the key-space. When dealing with multiple faulty cipher-texts, the Inbound phase is repeated to get a hyper-state for each faulty cipher-text and finally an element-wise intersection is taken over all the hyper-states. This intersection largely reduces the size of the final hyper-state. This reduced-size hyper-state is given as output of the Inbound phase. The Outbound phase proceeds as described above. The following algorithm presents the complete Escape attack while the pictorial description is given in Fig. 7. It is implied that the number of faults is inversely proportional to the size of the final key-search space. Extensive computer simulations using randomized diagonal faults have revealed that while a single fault can lead to a average reduced key-space of 2^{80}, using 2 faults the key-space can be reduced to as low as 2^{25} on an average. Increasing the number of faults to 4 helps to extract the exact key with a high probability.

procedure ESCAPE($c, \{c'_1, c'_2, \cdots, c'_n\}$) ▷ $n \rightarrow$ # of faulty outputs
  for $f = 1 : n$ **do**
    $s^{h_f} \leftarrow$INBOUND$(c \oplus c'_f)$ ▷ Compute hyper-state
  end for
  $s^h = \bigcap_{f=1}^{n} s^{h_f}$ ▷ Intersect all hyper-states
  $\mathcal{K}_{red}^{t^h} \leftarrow$OUTBOUND$(s^h, c)$
  $\mathbb{K} = \varnothing$
  for all $w \in \left(\bigtimes_{j=0}^{7} \mathcal{K}_{red}^{t^h_{*,j}} \right)$ **do**
    $k = \alpha_{12}(w) \oplus c$
    $\mathbb{K} = \mathbb{K} \cup \{k\}$
  end for
  return $\mathbb{K}$
end procedure

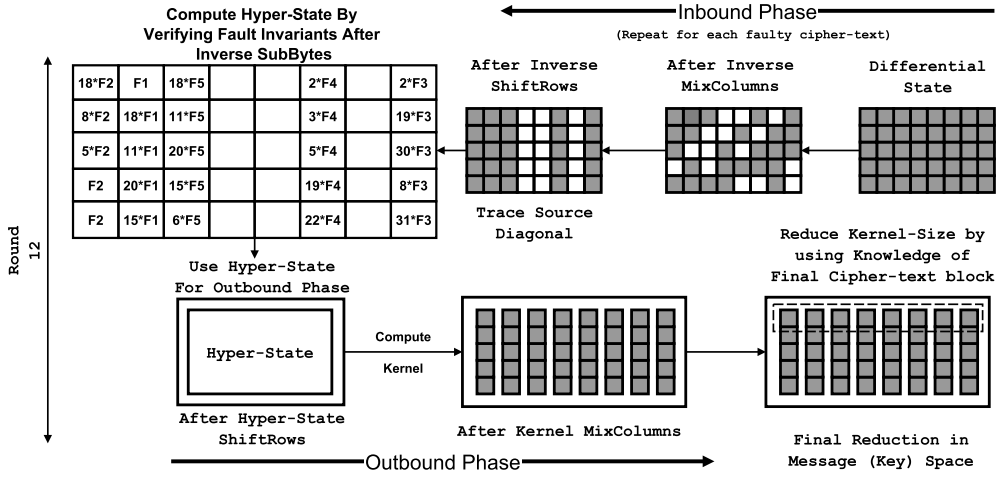

Fig. 7: The ESCAPE attack using a diagonal fault in d_1

5 Conclusion

We present arguments stating that the otherwise desirable feature of misuse-resistance can become the gateway for mounting differential faults attacks. Building upon this idea we develop an efficient diagonal fault attack on the APE authenticated cipher. APE is the first permutation based AE scheme which uses the Sponge based mode of operation. Hence the result presented here can be interpreted as the first fault analysis of Sponge when used in the context of authenticated encryption. The number of faulty cipher-texts required to reduce the key-search space to a practical limit is only 2 while it has been found that 4 faulty outputs can uniquely identify the key with a very high probability.

References

1. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Wang, Q., Yasuda, K.: PRIMATEs v1 (2014), http://competitions.cr.yp.to/round1/primatesv1.pdf
2. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Wang, Q., Yasuda, K.: PRIMATEs v1.01 (2014), http://primates.ae/wp-content/uploads/primatesv1.01.pdf
3. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: Lecture Notes in Computer Science, FSE. Springer-Verlag (2014), https://lirias.kuleuven.be/handle/123456789/450105
4. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and Authenticated Online Ciphers. In: ASIACRYPT (1). pp. 424–443 (2013)
5. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Cryptographic sponge functions, Available at http://sponge.noekeon.org/CSF-0.1.pdf
6. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: CRYPTO. pp. 513–525 (1997)
7. Bilgin, B., Bogdanov, A., Knezevic, M., Mendel, F., Wang, Q.: Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In: CHES. pp. 142–158 (2013)
8. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: EUROCRYPT. pp. 37–51 (1997)
9. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Eliminating Errors in Cryptographic Computations. J. Cryptology 14(2), 101–119 (2001)
10. Coron, J.S., Joux, A., Kizhvatov, I., Naccache, D., Paillier, P.: Fault Attacks on RSA Signatures with Partially Unknown Messages. In: Clavier, C., Gaj, K. (eds.) CHES. Lecture Notes in Computer Science, vol. 5747, pp. 444–456. Springer (2009)
11. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002)
12. Joye, M., Lenstra, A.K., jacques Quisquater, J.: Chinese Remaindering Based Cryptosystems in the Presence of Faults. Journal of Cryptology 12, 241–245 (1999)
13. Joye, M., Tunstall, M. (eds.): Fault Analysis in Cryptography. Information Security and Cryptography, Springer (2012), http://dblp.uni-trier.de/db/series/isc/isc364229655.html
14. Rogaway, P.: Nonce-Based Symmetric Encryption. In: FSE. pp. 348–359 (2004)
15. Saha, D., Mukhopadhyay, D., RoyChowdhury, D.: A Diagonal Fault Attack on the Advanced Encryption Standard. Cryptology ePrint Archive, Report 2009/581 (2009), http://eprint.iacr.org/