Kalpa Publications in Computing

Volume 11, 2019, Pages 49–63

Proceedings of 8th International Workshop
on Security Proofs for Embedded Systems

A comment on information leakage from robust code-based checkers detecting fault attacks on cryptographic primitives*

Osnat Keren[1] and Ilia Polian[2]

[1] Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel
osnat.keren@biu.ac.il
[2] Institute of Computer Engineering and Computer Architecture, University of Stuttgart, Germany
ilia.polian@informatik.uni-stuttgart.de

Abstract

Cryptographic hardware primitives must be protected against fault-injection attacks. Security-oriented error-detecting codes provide (probabilistic) guarantees for detection of maliciously injected faults even under assumption of a sophisticated attacker with access to powerful equipment. In this paper, we revisit the earlier finding that error-detection infrastructure may increase the undesired information leakage. We formalize the information leakage from the checker response by means of mutual information. We apply our analysis to the best security-oriented robust codes known today. We prove that the probability of an undetected attack is exponentially smaller than the entropy loss due to information leak from the checker. This means that an attack will be detected far before the attacker will gain significant information. Given a bound for acceptable information leakage (e.g., 0.5 bits of a 128-bit secret key), our analysis allows the designer to easily choose the number of redundant bits required to stay below that bound. The obtained results extend our knowledge about the relationship between detection capabilities of codes and information leakage due to them.

1 Introduction

Fault injection attacks on ciphers circuits aim to extract the secret key by analyzing correct and faulty ciphertext pairs. Such attacks grow in importance since novel cyber-physical applications and autonomous systems are physically exposed to and accessible by their users and can be manipulated. A large number of popular cryptosystems have been successfully attacked, ranging from simple block ciphers (e.g., AES [1], PRESENT [2], LED [3]) to elaborate asymmetric [4] and postquantum schemes [5, 6]. One of the most effective ways to mitigate fault injection attacks is by embedding security oriented codes in hardware. These codes are essential for mitigating faulty ciphertext-only attacks [7].

Security oriented codes can utilize deterministic or random encoders. The effectiveness of codes with random-encoding, e.g., the codes in [8, 9, 10, 11], depends on the entropy of the

*This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 923/16).

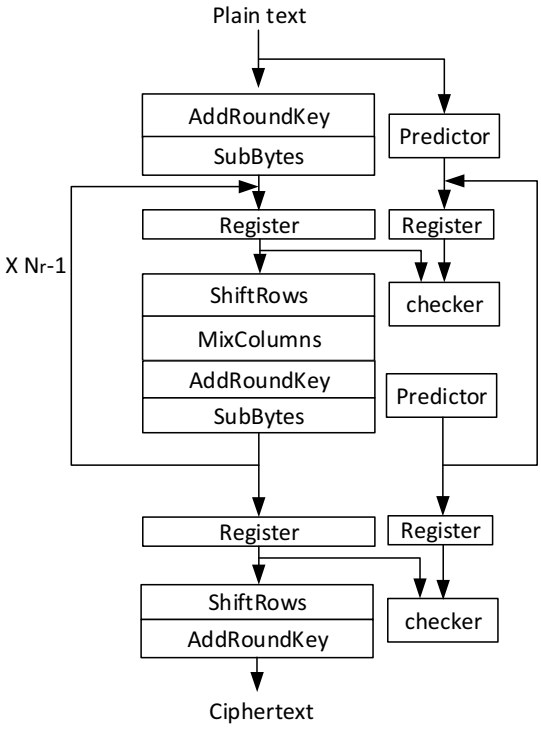

Figure 1: Protected AES architecture. The registers are located after the SubBytes module to enable on-line prediction and checking.

random portion. In practice, it is difficult and expensive to implement a true (i.e., maximal entropy) random number generator that cannot be neutralized by fault injection. This makes codes with deterministic encoding an attractive alternative [12]. In this paper we focus on information leakage from robust codes with deterministic encoding.

It is often argued that deterministic code-based countermeasures against fault injection attacks can increase the information leakage via the side channels (such as power consumption, electromagnetic radiation etc.). This claim is true to some extent; redundancy of any kind, as it is, carries information on the secret data it aims to protect. The question is how much exploitable information the redundant portion carries. In general, *duplication* based redundancy or *repeated computation* increase the SNR ratio (i.e., the ratio between the energy of the signal that carries information and the energy of the thermal and switching noises). It also increases the correlation between the secret information and the side information. In some cases, the security level can still degrade even if more complex codes are employed. In [13], the authors studied the effect of a *low rate* error detection circuits on resistance to power analysis attacks. The evaluation that was carried on a single S-box circuit showed that an adversary who knows the coding scheme can take advantage of this additional information. An analysis of the mutual information between the secret 8-bit key and the power trace showed that this attacker could learn at most 0.5 bits (depending on the noise level), whereas an attacker who was not aware of the implemented code had a lower success rate.

The observations presented in [13] apply to cases in which each SBox has its own redundant

Table 1: Information leakage (bits) and cumulative probability of an undetected (masked) fault attack for a 64-bit key protected by nine redundant bits

Number of faults d	1	2	3	4	16	251
Information leakage	0.25562	0.51124	0.76686	1.02248	4.08992	64
Masking probability	3.91E-03	1.53E-05	5,98E-08	2,34E-10	2,98E-39	$<$ 1E-604

bits; i.e., when the code rate is low. The AES architecture such as the one presented in [14] (see Fig 1) employs a predictor and a checker to protect all the state bits. The predictor computes the value of r redundant (check) bits in parallel to the execution. These bits, together with the 128 bits of the SubByte block (64 bits for lightweight ciphers), form a codeword. If a robust code is used, any attempt to tamper with the combinational logic or to flip the content of the registers will be detected by the checkers with a guaranteed probability. For example, a Compact Protection Code (CPC) [15] can protect any number of state bits with a worst-case probability of an undetected fault being 2^{-r+1} where r is the number of redundant bits. An attacker like the one in [13] who aims to extract only 8 state bits at a time cannot acquire information from the r redundant bits because they depend on the complete state and not only on its 8-bit portion; the relationship between 8 bits and the redundancy bits appears as totally random. In other words, the entropy of these eight bits does not decrease given the value of the r redundant bits.

Recently, several authors have observed that fault-tolerant digital design techniques affect SCA resistance [16, 17]. In [16] the authors proposed a combined single fault location attack on redundancy based countermeasures assisted by supplementary side channel information on the Hamming weight of the manifested error. However, this attack strategy applies to software implementations of AES and PRESENT ciphers that are protected by time/space redundancy. The type of redundancy used in these papers is not robust, and therefore the success of these attacks does not contradict our findings which assume security-oriented robust codes.

Contribution: In this paper we examine whether the use of robust codes with a deterministic encoder can degrade security; that is, whether the checker's response can narrow the key's search space. We evaluate the (average) reduction in the size of the search space in terms of the mutual information (MI) between the secret key and the checker's response. The MI is analyzed as a function of the injected error; it represents the (average) number of bits that can be learned about the key from a single injection of a specific error vector. We show that, when robust codes are used, the entropy loss due to information leak from the checker is small and increases linearly both in the number of redundant bits r, whereas the probability that the attack not be detected decreases exponentially. The same relationship holds for the dependency of the entropy loss and probability of an undetected attack on the number of fault injections. In other words, we can expect that providing strong fault detection has only a limited and exponentially decreasing impact on information leakage and thus on security.

For example, assume that a 64-bit secret key is protected with a security-oriented code with nine redundant bits. Table 2 provides one suitable code for this purpose. From our analysis it will follow that a standard checker leaks 0.25562 bits per fault injection and that the probability that a fault injection is undetected is $3.91 \cdot 10^{-3}$. Table 1 shows the total number of bits leaked after d fault injections and the probability that the attack will be detected. It can be seen that the number of attacks to extract a non-negligible amount of information is such that its detection is near to certain (e.g., masking probability of 10^{-30} for 16 fault injections necessary for extraction of 4 bits).

2 Preliminaries

2.1 Fault-injection attacks

It has been noticed in [18] that faults that occur during a cryptographic operation can compromise its security. Fault-injection attacks utilize this fact by deliberately inducing a physical disturbance during the execution of a cryptographic function. A plethora of fault-injection techniques has been demonstrated, including inducing glitches on the circuit's inputs or clock lines; overheating; underpowering; applying electromagnetic pulses; and illuminating the circuit with a laser [19, 20]. The best known attacks, which need a single fault injection to recover the complete secret key of a cipher [1, 3], require very high temporal and spatial resolution, i.e., a fault injection in a precisely known clock cycle at a precisely known portion of the circuit. Even the most sophisticated fault-injection techniques [21] currently do not provide such resolution, and as a consequence, an attack must be attempted several times. Robust codes used in this paper make a conservative assumption of the attacker: namely that the attacker can precisely select an arbitrary number of specific bits that will be flipped as a consequence of the fault injection. This is captured by defining the error as a binary vector which is added (XORed) to the circuit's outputs and errors which the attacker can choose.

Once a fault has been injected, the attacker can analyze the circuit's outputs (ciphertexts in the case of encryption) obtained in presence and in absence of the disturbance. Applying differential cryptanalysis techniques, secret key bits can be inferred. It is also possible to collect a larger number of the circuit's responses for faults of different intensities and to infer the key from a statistical analysis [22, 23]. In this paper, we make no assumption about which type of fault analysis the attacker will perform. Any injected fault that goes undetected by the robust code is considered critical and the codes aim at increasing the probability of detection. The question considered is whether knowing the checkbits gives the attacker additional information about the secret key and how much such information leakage can occur.

2.2 Robust codes based architectures

There is a fundamental difference between protecting the output of combinational logic (in our case, a cipher) and protecting data stored in memory arrays or registers. In the latter case the information portion is given explicitly, whereas in the first case, it is given implicitly. In other words, information about the secret key exists in the circuits' output as a function of functionality of the circuit, the user-controlled inputs and the internal state variables. In protected memory arrays, the codeword is stored in memory, whereas in protected cryptographic modules a (systematic) codeword is formed on the inputs of the checker.

When security-oriented codes are applied to hardware implementations of cryptographic primitives (e.g., block ciphers), they can interpret the entire state of the cipher (128 bits for AES-128, 64 bits for LED-64) as the information portion and add redundant bits according to the code structure. Detailed security-oriented code-based architectures are discussed in [24] for robust codes with and without error *correction* capabilities. Example 1 further below illustrates the application for the reduced example of a 12-bit cipher state being protected by two redundant bits.

Formally, denote by n_b the number of wires that enter the checker. We refer to this n_b-bit binary vector as a q-ary word of length $n = n_b/m$ symbols over an alphabet of size $q = 2^m$. In a fault-free circuit, the q-ary vector is a codeword that consists of k q-ary symbols which represent the output of the *original component*, and r q-ary redundant symbols generated by the predictor.

A schematic description of a circuit that implements a protected AES cipher is shown in Fig. 1. The inputs to the circuit are the plaintext s and a key y. We assume that s is known to the attacker and that the random variable Y that represents the key has maximal entropy; i.e., $H(Y) = m \cdot k$. In a fault-free scenario, the original component computes $x = f(s+y) \in \mathbb{F}_q^k$ in the first cycle and stores it in the register. From the point of view of the attacker x is a uniformly distributed random variable.

The plaintext s and the random key y also enter a predictor block that computes r redundant q-ary symbols. Denote by w the output of the predictor. w is a deterministic function of $s + y$. Since f is a bijective function, w.l.o.g. we can refer to w as a function of x. That is, $w(x) = w(f(s + y)) \in \mathbb{F}_q^r$. The pair $c = (x, w(x))$ forms a codeword of length $n = k + r$ in a systematic q-ary code $\mathcal{C}$ which is stored in the register.

An attacker may tamper with the original circuit and/or the predictor. Denote by $\hat{c}$ the word on the checker's inputs. In a fault-free circuit $\hat{c} = c$. The difference between $\hat{c}$ and the correct codeword c is denoted by e. That is, $e = \hat{c} - c \in \mathbb{F}_q^n$ is an additive error vector. Experimental results reported in [25], show that some fault injection techniques, e.g. by jittering the clock, causes arbitrary number of bit flips. These bit flips are modeled as additive errors over a finite field of characteristic 2.

The checker examines the word $\hat{c}$ it sees on its inputs. If it decides that the word is not a legal codeword or that it cannot be corrected into a legal one, it outputs the value $\perp$. Depending on the design, a $\perp$ may halt the processor or cause the system to replace the output by a random one. If the checker decides that the word is legal or that it is within the correction capability of the code, it decodes it into a legal codeword. The checker's output is denoted by v, $v \in \mathcal{C} \cup \{\perp\}$. If $v \in \mathcal{C}$, it is of the form $v = (v_x, v_w)$.

The same takes place in the following cycles; the register is loaded with a codeword $c \in \mathcal{C}$ computed by the predictor. However, here s is not the plaintext, but rather a deterministic function of the (key dependent) state bits of the register.

Since the checker is implemented in hardware on the same circuit as the logic which it protects, the checker itself can be subject to fault injection as well. Usually, the checker is composed of two parts: an encoder and a comparator. An attack on the encoder logic is equivalent to an attack on the register and hence will be detected. Detection of an attack on the comparator requires a security aware implementation [26].

Note that the data processed by the last two blocks in Fig. 1 are not protected. An effective attack on these blocks will aim to stick the bits and not flip them. Thus, codes detecting stuck-at faults will be more effective. This however is beyond of the scope of this paper. A detailed analysis of the security provided by unidirectional error detecting codes can be found in [27].

2.3 Effectiveness of security oriented codes

In general, the effectiveness of a code as a countermeasure against fault injection attacks is measured in terms of its error detection capability. A code $\mathcal{C}$ is said to be *robust* if it can detect any additive error $e = (e_x, e_w) \in \mathbb{F}_q^n$ with non zero probability. Specifically, under a uniform distribution of x, the probability $Q(e)$ that a nonzero error e is masked by the codewords is

$$Q(e) = q^{-k} |\{c \ : \ c, c + e \in \mathcal{C}\}| < 1.$$

The code's error masking probability is defined as $\bar{Q} = \max_{e \neq 0} Q(e)$. The value of $\bar{Q}$ is lower bounded by the average error masking probability; namely, $\bar{Q} \geq 2^{-r \cdot m}$. Codes that fulfill this bound on equality are called optimum robust codes. There are very few high rate optimum or close to optimum robust error detecting codes; most are variants of two basic structures:

the Quadratic-Sum (QS) code [28] and Punctured-Square (PS)/ Punctured-Cubic (PC) codes [29, 30]. Punctured codes exist for any (k, r) pair but their implementation cost is high, whereas the QS based codes have low implementation costs but impose restrictions on the relationship between k and r. The best robust code known so far is the compact protection code (CPC), which provides a high rate and low implementation cost, and exists for any k and r [15].

In the next section, we examine how many information bits leak from a checker that utilizes robust codes.

3 Information leakage from robust code based checkers

We assume that the attacker knows the code and hence can know which errors will be always detected ($Q(e) = 0$) and which have a chance to pass unnoticed ($Q(e) > 0$). Therefore, from here on, we assume that an error e of the latter type is injected. We measure the information that leaks from the checker in terms of the mutual information between two random variables: the key Y and the checker output V.

Formally [31], let Z be a discrete random variable defined with a support set $\mathcal{Z}$ and denote by $p(z) = P_Z(z)$ the probability that Z will take the value z. Similarly, for two random variables $Z \in \mathcal{Z}, W \in \mathcal{W}$ denote by $p(z|w) = P_{Z|W}(z|w)$ the conditional probability that $Z = z$ given $W = w$. The entropy of a random variable Z is defined as

$$H(Z) \triangleq -\sum_{z\in\mathcal{Z}} p(z)\log_2(p(z)),$$

and the conditional entropy of Z given W is

$$H(Z|W) \triangleq -\sum_{z\in\mathcal{Z},w\in\mathcal{W}} p(z,w)\log_2(p(z|w)).$$

The mutual information between two random variables Z and W is

$$I(Z;W) \triangleq \sum_{z\in\mathcal{Z},w\in\mathcal{W}} p(z,w)\log_2\left(\frac{p(z,w)}{p(z)p(w)}\right) = H(Z) - H(Z|W).$$

Using these notations, the conditional mutual information between Y and V for a given error e and a chosen plaintext s is

$$I(Y;V|s,e) = \sum_{y,v\in\mathbb{F}_2^{mk}} p(y,v|s,e)\log_2\left(\frac{p(y,v|s,e)}{p(y,v|s,e)p(y,v|s,e)}\right).$$

Similarly, if the plaintext is not chosen by the attacker, then the mutual information is

$$I(Y;V|S,e) = \sum_{s\in\mathbb{F}_2^{mk}} p(s)I(Y;V|s,e).$$

In this section we consider two attack scenarios: an attack on the first round and an attack on the i-th round for $i > 1$. In general, fault analysis attacks are performed on the middle/last rounds because of the trade-off between the number of required correct-fault pairs and the time-memory complexity [32, 33]. Nevertheless, when error detecting codes are embedded in hardware, an attacker who knows the coding scheme can derive information about the key even when the attack is performed on the very first cycle. The following example illustrates how the checker's output may be used to reduce the search space of the key:

Example 1. *For simplicity, consider a hypothetical 12-bit cipher. The inputs to the circuit implementing that cipher are a 12-bit plaintext s and a 12-bit key y. Assume that in the first cycle this cipher computes*

$$x = (x_1, x_2, \dots x_{12}) = f(s+y) = \begin{cases} (s+y)^{-1} & s+y \neq 0 \\ 0 & s+y = 0 \end{cases}$$

where the computation is performed in $\mathbb{F}_{2^{12}}$ with the primitive polynomial $D^{12}+D^6+D^4+D+1$. Consider a hardware implementation protected by the Quadratic-Sum (QS) code. The block we are referring to as "original component" calculates $x(s,y)$, and in parallel, the predictor generates two redundant bits $w_1(s,y)$ and $w_2(s,y)$. When the two binary vectors x and w are treated as vectors over $\mathbb{F}_2^2$, i.e., $m=2, \xi_i = (x_{2i-1}, x_{2i})$ and $\eta = (w_1, w_2)$, they form a codeword that fulfills the following property:

$$\eta = \xi_1\xi_2 + \xi_3\xi_4 + \xi_5\xi_6. \tag{1}$$

The multiplication and additions in Eq. 1 are over $\mathbb{F}_4$ with the primitive polynomial D^2+D+1. ($\mathbb{F}_4$ is isomorphic to $\mathbb{F}_2^2$). Note that the predictor generates the two redundant bits directly from s and y without explicitly computing the variable x, even though η and ξ fulfill Eq. 1. The codeword stored in the register is then $c = (x, w)$. Our attack model assumes that the adversary can flip an arbitrary number of bits in arbitrary locations of both x and w.

Assume that $s =$ (00 01 11 11 01 11), $y =$ (00 00 11 10 10 10) and the attacker injects the error $e =$ (01 00 00 00 00 10, 11). Then, the codeword written into the register is

$$c = (\underbrace{11\ 00\ 11\ 01\ 10\ 00}_{x}, \underbrace{11}_{w}) = (\underbrace{3\ 0\ 3\ 1\ 2\ 0}_{\xi}, \underbrace{3}_{\eta})$$

and the distorted word $\hat{c} = c + e = (\hat{x}, \hat{w}) = (\hat{\xi}, \hat{\eta})$ read from the register is

$$\hat{c} = c + e = (\mathbf{1}0\ 00\ 11\ 01\ 10\ \mathbf{1}0, \mathbf{00}) = (\underbrace{\mathbf{2}\ 0\ 3\ 1\ 2\ \mathbf{2}}_{\hat{\xi}}, \underbrace{\mathbf{0}}_{\hat{\eta}})$$

(the erroneous bits and symbols appear in bold). The checker computes $\hat{\xi}_1\hat{\xi}_2 + \hat{\xi}_3\hat{\xi}_4 + \hat{\xi}_5\hat{\xi}_6 = 0$ and since it is equal to $\hat{\eta}$, it does not report a decoding error. The attacker who knows the code sees that the attack has been masked (undetected). Therefore, the attacker infers that the following error-masking equation must be fulfilled:

$$(\xi_1 + 1)\xi_2 + \xi_3\xi_4 + \xi_5(\xi_6 + 2) = \eta + 3.$$

There are exactly 2^{10} values of x that satisfy this equation; they are of the form

$$\xi = (\xi_1,\ \xi_2 = 3 - 2\xi_5,\ \xi_3,\ \xi_4,\ \xi_5,\ \xi_6).$$

Since $y = x^{-1} - s$), y can take 2^{10} values. Therefore, the search space for the key y has been narrowed from 12 to 10 bits, or by 2 bits.

Note that if the value of the key would have been such that the error would have been detected, the attacker could observe this outcome and narrow the search space for y from 2^{12} down to $(2^{12} - 2^{10})$ possible values. However, the system would then receive an alarm and prevent the attacker from continuing to reduce the search space by further fault injections. For example, the system could initiate re-keying, in which case all knowledge that the attacker had gained previously becomes useless. Note that the error is not detected only for $Q(e) = 1/4$ of the keys for this error vector due to the properties of the used QS code. It is detected (leading to an alarm) for the remaining $3/4$ of the keys. □

In what follows we prove that on average the information leakage is small. An attacker can observe the ciphertext, and hence he can determine whether the checker has outputed a different v. Moreover, the cipher output is a deterministic function of v. Therefore, we assume that v is observable by the attacker.

3.1 Fault attack on the first round

We distinguish between standard checkers that output a $\perp$ symbol when the word on their input is illegal, and infective checkers that output a random codeword upon detecting such a problem. The theorem below states that the information that can be extracted from the checker response in the presence of an error decreases exponentially with the number of redundant bits.

Theorem 1. *Let $\mathcal{C}$ be a systematic q-ary robust code of cardinality q^k, length $n = k + r$, and an error masking probability $Q(e)$. The mutual information (in bits) between the secret key y and a standard checker output $v \in \mathbb{F}_q^n$ for a given plaintext $s \in \mathbb{F}_q^k$ and an injected error $e \in \mathbb{F}_q^n$ is*

$$\begin{aligned} I_{\text{standard}}(Y;V|s,e) &= mkQ(e) \\ &\quad -(1-Q(e)) \cdot \log_2(1-Q(e)). \end{aligned} \tag{2}$$

The mutual information between y and an infective checker output v for a given plaintext s and an injected error e is

$$\begin{aligned} I_{\text{infective}}(Y;V|s,e) &= mkQ(e) \\ &\quad -Q(e)(2-Q(e))\log_2(2-Q(e)) \\ &\quad -(1-Q(e))^2\log_2(1-Q(e)). \end{aligned} \tag{3}$$

Proof. **Information leakage from standard checkers:**
The secret key $y \in \mathbb{F}_q^k$ is a uniformly distributed random variable; hence, so is $x = f(s,y)$ where f is a cryptographic function and thus bijective. Denote by $\mathcal{M}_e$ the set of x's associated with the codewords that mask the error e; the size of this set is $|\mathcal{M}_e| = q^k Q(e)$. The conditional probability distribution of the checker output is

$$\begin{aligned} P_V(v|e, x \in \mathcal{M}_e) &= \begin{cases} 1 & v \in \mathcal{C}, v_x = x + e_x \\ 0 & \text{otherwise} \end{cases} \\ P_V(v|e, x \notin \mathcal{M}_e) &= \begin{cases} 1 & v = \perp \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Thus,

$$P_V(v|e) = \begin{cases} q^{-k} & v \in \mathcal{C}, v_x \in e_x + \mathcal{M}_e \\ 1 - Q(e) & v = \perp \\ 0 & \text{otherwise} \end{cases}$$

Under the assumption that the attacker knows the codebook and can choose s, it is clear that when the checker raises a $\perp$ flag it provides information about the key y. The mutual

information between y and the checker output v for a given plaintext s and an injected error e is

$$\begin{aligned} I_{\text{standard}}(Y;V|s,e) &=_{(a)} I_{\text{standard}}(X;V|e) \\ &= H(V|e) - H(V|X,e) \\ &=_{(b)} H(V|e). \end{aligned}$$

The correctness of this equality follows from: a) v and s are statistically independent random variables, and b) $H(V|X,e) = 0$ since the code is deterministic. The probability of $\perp$ symbol for a given error vector is $1 - Q(e)$; thus, for robust codes we have,

$$H(V|e) = Q(e) \cdot \log_2(q^k) - (1 - Q(e)) \cdot \log_2(1 - Q(e)).$$

Information leakage from infective checkers:
Infective checkers output a random symbol upon detecting an error. Therefore we have,

$$P_{V,X}(v,x|e) = \begin{cases} q^{-k} & v_x = e_x + x, x \in \mathcal{M}_e \\ q^{-2k} & x \notin \mathcal{M}_e \\ 0 & \text{otherwise} \end{cases},$$

and since $q^{-k}|\mathcal{M}_e| = Q(e)$ we have,

$$P_V(v|e) = \begin{cases} q^{-k}(2 - Q(e)) & v_x \in e_x + \mathcal{M}_e \\ q^{-k}(1 - Q(e)) & v_x \notin e_x + \mathcal{M}_e \\ 0 & v = \perp \end{cases}.$$

The conditional entropy of V is

$$\begin{aligned} H(V|e) = {} & mk - Q(e)(2 - Q(e)) \log_2(2 - Q(e)) \\ & -(1 - Q(e))^2 \log_2(1 - Q(e)) \end{aligned}$$

and

$$H(V|X,e) = mk(1 - Q(e)) + \underbrace{q^{-k}|\mathcal{M}_e| \cdot 1 \log_2(1)}_{=0}.$$

The mutual information between y and the checker output for a given error for an infective checker is then

$$I_{\text{infective}}(Y;V|s,e) = I_{\text{infective}}(X;V|e) = H(V|e) - H(V|X,e).$$

□

Note that optimum (or close to optimum) robust codes have $Q(e) \leq 2^{-mr+1}$ [28]; therefore, both $I_{\text{standard}}(Y;V|s,e)$ and $I_{\text{infective}}(Y;V|s,e)$ are smaller than $mk2^{-mr+1}$. Hence, for a given key size and a bound for acceptable information leakage $I_{\max}$ (per injection), a code designer has to use at least $\lceil \log_2(mk/I_{\max}) \rceil + 1$ redundant bits to stay below this bound.

Denote by $\bar{I}_{\text{standard}}$ and $\bar{I}_{\text{infective}}$ the maximal mutual information $I(Y;V|s,e)$ over all the non-zero errors e and plaintexts s. Tables 2 and 3 show the maximal error masking probability (EMP) and the information leakage for different q-ary compact protection codes (CPCs) suitable

for protecting 64- and 128-bit ciphers, respectively. The columns of the table are as follows. The first column is the number of redundant bits $m \cdot r$. The second column shows, for each number of redundant bits, one possible construction of a CPC (explained further below). The EMP of each suggested CPC, $\bar{Q}$, computed as explained in [15] appears in column 3. The maximal number of bits that can be learned by observing the output of a standard and infective checker appears in columns 4 and 5, respectively. These values have been calculated using Eqs. 2 and 3.

In general, a CPC is constructed from $l \geq 1$ ground codes by concatenating their information portions and XORing their redundant portions. The ground codes $\mathcal{C}(\hat{n}_i, \hat{k}_i, \hat{m}_i)$, $i = 1, \ldots l$ are systematic robust codes of length $\hat{n}_i$ bits and size $2^{\hat{k}_i}$ defined over the finite field $\mathbb{F}_{2^{\hat{m}_i}}$. Here, we ignore the implementation cost of the codes and use (simple to describe) constructions that provide minimal error masking probability. For *optimal* constructions with *minimal implementation cost* refer to [15]. The acronyms used in Tables 2 and 3 are the following: QS stands for the Quadratic-Sum code [28], PC stands for Punctured-Cubic [30], and sQS and TQS for shortened QS and triple QS codes [15]. The following example explains one specific CPC from Table 2 and how the checker operates for that code.

Example 2. *Consider a 64-bit key, protected by a CPC with five redundant bits ($m \cdot r = 5$). There are several ways to construct such a code, the simplest one is shown in Table 2. The code is a composition of two ground codes: a QS(55,50,5) code and a PC(19,14,14) code. This means that the checker treats $\hat{x}$, the 64-bit vector at its input, as a mixed-alphabet vector composed of ten 5-bit symbols $\hat{\xi}_1, \ldots, \hat{\xi}_{10}$ from the finite field $\mathbb{F}_{2^5}$ and a single 14-bit symbol $\hat{\xi}_{11}$ from $\mathbb{F}_{2^{14}}$*

$$\hat{x} = (\underbrace{\hat{x}_1, \ldots, \hat{x}_5}_{\hat{\xi}_1}, \underbrace{\hat{x}_6, \ldots \hat{x}_{10}}_{\hat{\xi}_2}, \ \ldots \ , \underbrace{\hat{x}_{46}, \ldots, \hat{x}_{50}}_{\hat{\xi}_{10}}, \underbrace{\hat{x}_{51}, \ldots \hat{x}_{64}}_{\hat{\xi}_{11}}).$$

Note that the error masking of a CPC is determined by the weakest ground code [15]. In our case $\bar{Q}_{\mathrm{QS}} = 2^{-5}$ and $\bar{Q}_{\mathrm{PC}} = 2^{-4}$, thus the error masking probability of the composed code is $2^{-4} = 0.0625$ (refer to Table 2).

Recall that x is the correct vector that was written into the register and $\hat{x}$ is the vector read from the register. In case of fault $\hat{x}$ may be different from x. The checker computes

$$\begin{aligned} \hat{\eta}_1 &= \hat{\xi}_1 \cdot \hat{\xi}_2 + \hat{\xi}_3 \cdot \hat{\xi}_4 + \cdots \hat{\xi}_9 \cdot \hat{\xi}_{10}, && \text{for the QS(55,50,5) code} \\ \hat{\eta}_2 &= (\hat{\xi}_{11})^3 \cdot P_{14\times 5}, && \text{for the PC(19,14,14) code} \end{aligned}$$

(In this example, $P_{14\times 5}$ is a binary matrix of rank 5 that had been chosen arbitrarily). Then, the checker computes $\eta = \hat{\eta}_1 + \hat{\eta}_2$ and compares it with the predicted value $\hat{w}$ read from the 5-bit register. In the case of a fault, $\hat{w}$ may differ from w. If the computed value differs from $\hat{w}$, the checker outputs the symbol $\perp$, which will trigger an alarm.

Note that this construction is not the cost-optimal CPC because the five redundant bits of the QS code are computed from 50 information bits by multiplications over a small finite field ($\mathbb{F}_{2^5}$), whereas the five redundant bits of the PC code (to be XORed with the five bits of the QS code) are computed from the remaining 14 (= 64 − 50) information bits by multiplications over a large field ($\mathbb{F}_{2^{14}}$). Optimal and close to optimal constructions are given in [15]. □

It is interesting to compare the leakages from standard and infective checkers in Tables 2 and 3. Intuitively, one would expect that the infective checker leaks considerably less information that its counterpart, because it never generates the special symbol $\perp$ and aims at preventing the adversary from receiving ciphertexts that are suitable for extracting information. The results suggest that there is a difference between the information leakages from the standard and the

Table 2: Error masking probability and information leakage from the checker (in bits) for 64bit key

# redundant bits (mr)	CPC	EMP $\bar{Q}$	Standard checker $\bar{I}_{\text{standard}}$	Infective checker $\bar{I}_{\text{infective}}$
2	QS(66,64,2)	2.50E-01	16.31128	15.88024
3	QS(63,60,3)+PC(7,4,4)	2.50E-01	16.31128	15.88024
4	QS(68,64,4)	6.25E-02	4.08729	3.96629
5	QS(55,50,5)+PC(19,14,14)	6.25E-02	4.08729	3.96629
6	TQS(60,54,6)+PC(16,10,10)	3.13E-02	2.04437	1.98286
7	QS(63,56,7)+ PC(15,8,8)	1.56E-02	1.02237	0.99136
8	QS(72,64,8)	3.91E-03	0.25562	0.24783
9	QS(63,54,9) + PC(19,10,10)	3.91E-03	0.25562	0.24783
10	TQS(60,50,10)+ PC(24,14,14)	1.95E-03	0.12782	0.12391
11	sQS(75,64,11)	9.77E-04	0.06391	0.06196
12	QS(60,48,12) + PC(28,16,16)	4.88E-04	0.03195	0.03098
13	TQS(52,39,13) + sQS(38,25,13)	2.44E-04	0.01598	0.01549
14	4 × PC(30,16,16)	1.22E-04	0.00799	0.00774
15	4 × PC(31,16,16)	6.10E-05	0.00399	0.00387
16	QS(80,64,16)	1.53E-05	0.00100	0.00097

Table 3: Error masking probability and information leakage from the checker (in bits) for 128bit key

# redundant bits (mr)	CPC	EMP $\bar{Q}$	Standard checker $\bar{I}_{\text{standard}}$	Infective checker $\bar{I}_{\text{infective}}$
2	QS(130,128,2)	2.50E-01	32.31128	31.88024
3	QS(126,123,3) + PC(8,5,5)	2.50E-01	32.31128	31.88024
4	QS(132,128,4)	6.25E-02	8.08729	7.96629
5	QS(125,120,5) + PC(13,8,8)	6.25E-02	8.08729	7.96629
6	QS(126,120,6) + PC(14,8,8)	3.13E-02	4.04437	3.98286
7	TQS(126,119,7)+ PC(16,9,9)	1.56E-02	2.02237	1.99136
8	QS(136,128,8)	3.91E-03	0.50562	0.49783
9	QS(117,108,9)+ 2 × PC(19,10,10)	3.91E-03	0.50562	0.49783
10	TQS(110,100,10) + 2 × PC(24,14,14)	1.95E-03	0.25282	0.24891
11	QS(99,88,11)+ QS(50,40,10)	9.77E-04	0.12641	0.12446
12	QS(108,96,12) + 2 × PC(28,16,16)	4.88E-04	0.06320	0.06223
13	QS(117,104,13) + QS(36,24,12)	2.44E-04	0.03160	0.03111
14	QS(126,112,14) + PC(30,16,16)	1.22E-04	0.01580	0.01556
15	QS(126,112,14) + PC(31,16,16)	6.10E-05	0.00790	0.00778
16	QS(144,128,16)	1.53E-05	0.00198	0.00194

infective checkers for all considered CPCs, but that this difference is not that large. This can be explained by the low error-masking probability of CPCs, which are optimal or close to optimal by construction. Therefore, the term $mkQ(e) = mk \cdot 2^{-rm}$, which decreases exponentially in r, dominates the expressions in Eqs. 2 and 3.

3.2 Fault attack on round $i > 1$

When attacking one of the internal rounds of a cipher, the attacker does not know the round state s_i and the round key y_i, but does know the functionality and can choose the error vector. Therefore, an indication that the error has been detected provides information about the subset (or subspace) that current $s_i + y_i$ belongs to. Formally, assume that no information is gained on the key prior to the fault injection on round i. This implies that for uniformly distributed key, the round state s_i is also uniformly distributed. Hence, the information that can be learned is

$I(Y_i; V_i|S, e) = I(Y_i; V_i|s_i, e)$. In fact, the attacker needs to have hypothesized on the previous $i-1$ round keys in order to make use of this information on y_i. In other words, it is harder to extract information from an attack on the i'th round than from attack on the first round.

Consider now a different scenario in which the attacker has narrowed the key's search space by ν bits prior to the attack. According to our assumption, the attacker knows the code and thus can *choose* an error e of minimal error detecting probability. (Note that we ignore the fact that this cannot be done in polynomial time). In this case, the probability $Q^{(\nu)}(e)$ that a code will mask (i.e., not detect) e given that only $2^{mk-\nu}$ out of its $2^{mk} = q^k$ codewords are equally likely to appear is upper bounded by

$$Q^{(\nu)}(e) \leq \frac{q^k \bar{Q}}{2^{mk-\nu}} = 2^{\nu} \bar{Q}. \tag{4}$$

This implies that if the number of redundant bits is smaller than ν, an attacker whose computational resources are (almost) unlimited may be able to inject a fault without being notices. The following example illustrates the computational effort involved in using the ν bits discovered so far to extract significant information about the remaining $(mk - \nu)$ secret bits.

Example 3. *Consider the CPC in Ex. 2 which has five redundant bits. Assume the attacker knows the first $\nu = 7$ bits of the key. Since $7 > 5$, it follows from Eq. 4 that it may be possible to find an error e that will be masked by the codeword stored in the register. By injecting this error (if it exists) the attacker will be able to conduct a fault analysis attack without being noticed.*

Let $\mathcal{A}$ denote a subset of size $2^{57} = 2^{64-\nu}$ that contains all the possible values that the x may take at the i'th round. Assume that the attacker has the resources (processing power and memory space) to construct $\mathcal{A}$. Given $\mathcal{A}$, the attacker has to choose an error vector

$$e = (e_1, \ldots, e_{64}, e_{65}, \ldots e_{69}) = (\epsilon_1, \ldots, \epsilon_{10}, \epsilon_{11}, \epsilon_{12})$$

of maximal $Q^{(\nu)}(e)$. For doing that, for each e (that has not been injected prior to the current injection) the attacker has compute the size of $\mathcal{A} \cap \mathcal{B}$ where $\mathcal{B}$ is the subset of codewords (ξ, η) that satisfy the equation

$$\eta + \epsilon_{12} \quad = \quad (\xi_1 + \epsilon_1) \cdot (\xi_2 + \epsilon_2) + \cdots (\xi_9 + \epsilon_9) \cdot (\xi_{10} + \epsilon_{10}) + (\xi_{11} + \epsilon_{11})^3 \cdot P_{14\times 5}$$

where $\epsilon_i \in \mathbb{F}_2^5$ for $i \neq 11$ and $\epsilon_{11} \in \mathbb{F}_2^{14}$. □

Clearly the complexity of calculating the size of $\mathcal{A} \cap \mathcal{B}$, even for a single e, is as large as the complexity of a brute force search of the unknown $(64 - \nu)$ key bits. In fact, there is no viable (i.e., low-complexity) way in which the attacker can make use of the known ν key bits in order to increase the chances of not being detected by a CPC checker. Therefore, there seems to be no practical way for the attacker to use the additional knowledge of key bits for injection of faults that will lead to a larger information leakage than in the first round.

4 Conclusions

One of the most effective ways to mitigate fault injection attacks is by embedding security oriented codes in hardware. In this paper, we examined whether the use of robust codes with a deterministic encoder can degrade security. We formalized the information leakage from the checker response by means of mutual information. We proved that the probability that an attack will go undetected is exponentially smaller than the entropy loss due to information leak from the checker. Given a bound for acceptable information leakage, our analysis allows the designer to easily choose the number of redundant bits required to stay below that bound.

References

[1] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," in *WISTP*, vol. 6633 of *Lecture Notes in Computer Science*, pp. 224–233, Springer, 2011.

[2] F. D. Santis, O. M. Guillen, E. Sakic, and G. Sigl, "Ciphertext-only fault attacks on PRESENT," in *LightSec*, vol. 8898 of *Lecture Notes in Computer Science*, pp. 85–108, Springer, 2014.

[3] P. Jovanovic, M. Kreuzer, and I. Polian, "A fault attack on the LED block cipher," in *COSADE*, vol. 7275 of *Lecture Notes in Computer Science*, pp. 120–134, Springer, 2012.

[4] A. Alkhoraidly, A. Dominguez-Oviedo, and M. A. Hasan, "Fault attacks on elliptic curve cryptosystems," in *Fault Analysis in Cryptography*, Information Security and Cryptography, pp. 137–155, Springer, 2012.

[5] J. Krämer and M. Loiero, "Fault attacks on UOV and rainbow," in *COSADE*, vol. 11421 of *Lecture Notes in Computer Science*, pp. 193–214, Springer, 2019.

[6] F. Valencia, I. Polian, and F. Regazzoni, "Fault sensitivity analysis of lattice-based post-quantum cryptographic components," in *SAMOS*, 2019. (accepted for publication).

[7] T. Fuhr, E. Jaulmes, V. Lomné, and A. Thillard, "Fault attacks on aes with faulty ciphertexts only," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 108–118, 2013.

[8] R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs, "Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors," in *Advances in Cryptology–EUROCRYPT 2008*, pp. 471–488, Springer, 2008.

[9] Z. Wang and M. Karpovsky, "Algebraic manipulation detection codes and their applications for design of secure cryptographic devices," in *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*, pp. 234–239, IEEE, 2011.

[10] X. T. Ngo, S. Bhasin, J. Danger, S. Guilley, and Z. Najm, "Linear complementary dual code improvement to strengthen encoded circuit against hardware trojan horses," in *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pp. 82–87, 2015.

[11] S. Dziembowski, K. Pietrzak, and D. Wichs, "Non-malleable codes." Cryptology ePrint Archive, Report 2009/608, 2009. http://eprint.iacr.org/2009/608.

[12] O. Keren and M. Karpovsky, "Relations between the entropy of a source and the error masking probability for security-oriented codes," *IEEE Transactions on Communications*, vol. 63, no. 1, pp. 206–214, 2015.

[13] F. Regazzoni, L. Breveglieri, P. Ienne, and I. Koren, "Interaction between fault attack countermeasures and the resistance against power analysis attacks," in *Fault Analysis in Cryptography*, 2012.

[14] B. Karp, M. Gay, O. Keren, and I. Polian, "Security-oriented code-based architectures for mitigating fault attacks," in *DCIS*, pp. 1–6, IEEE, 2018.

[15] H. Rabii, Y. Neumeier, and O. Keren, "High rate robust codes with low implementation complexity," *IEEE Transactions on Dependable and Secure Computing, DOI: 10.1109/TDSC.2018.2816638*, 2018.

[16] S. Saha, D. Jap, J. Breier, S. Bhasin, D. Mukhopadhyay, and P. Dasgupta, "Breaking redundancy-based countermeasures with random faults and power side channel," in *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 15–22, 2018.

[17] J. Riha, V. Miskovský, H. Kubatova, and M. Novotný, "Influence of fault-tolerance techniques on power-analysis resistance of cryptographic design," *2017 Euromicro Conference on Digital System Design (DSD)*, pp. 260–267, 2017.

[18] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of eliminating errors in cryptographic computations," *J. Cryptology*, vol. 14, no. 2, pp. 101–119, 2001.

[19] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[20] I. Polian and F. Regazzoni, "Counteracting malicious faults in cryptographic circuits," in *ETS*, pp. 1–10, IEEE, 2017.

[21] S. Tajik, H. Lohrke, F. Ganji, J. Seifert, and C. Boit, "Laser fault attack on physically unclonable functions," in *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 85–96, 2015.

[22] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis," in *CHES*, vol. 6225 of *Lecture Notes in Computer Science*, pp. 320–334, Springer, 2010.

[23] N. F. Ghalaty, B. Yuce, M. M. I. Taha, and P. Schaumont, "Differential fault intensity analysis," in *FDTC*, pp. 49–58, IEEE Computer Society, 2014.

[24] B. Karp, M. Gay, O. Keren, and I. Polian, "Security-oriented code-based architectures for mitigating fault attacks," in *2018 Conference on Design of Circuits and Integrated Systems (DCIS)*, pp. 1–6, 2018.

[25] B. Karp, M. Gay, O. Keren, and I. Polian, "Detection and correction of malicious and natural faults in cryptographic modules," in *PROOFS 2018, 7th International Workshop on Security Proofs for Embedded Systems*, pp. 68–82, 2018.

[26] F. Busaba, P. K. Lala, and A. Walker, "On self-checking design of CMOS circuits for multiple faults," *VLSI Design*, vol. 1998, no. 2, pp. 151–161, 1998.

[27] A. Burg and O. Keren, "On the efficiency of berger codes against error injection attacks on parallel asynchronous communication channels," *Information Security Journal: A Global Perspective*, vol. 22, no. 5-6, pp. 208–215, 2013.

[28] M. G. Karpovsky, K. J. Kulikowski, and Z. Wang, "Robust error detection in communication and computational channels," in *Spectral Methods and Multirate Signal Processing. SMMSP'2007. 2007 International Workshop on*, Citeseer, 2007.

[29] N. Admaty, S. Litsyn, and O. Keren, "Puncturing, expurgating and expanding the q-ary bch based robust codes," in *Electrical Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of*, pp. 1–5, Nov 2012.

[30] Y. Neumeier and O. Keren, "Robust generalized punctured cubic codes," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2813–2822, 2014.

[31] R. Gallager and L. Laboratory, *Sequential Decoding for Binary Channels with Noise and Synchronization Errors*. Group report, Massachusetts Institute of Technology, Lincoln Laboratory, 1961.

[32] M. Rivain, "Differential fault analysis on des middle rounds," in *CHES*, 2009.

[33] P. Derbez, P.-A. Fouque, and D. Leresteux, "Meet-in-the-middle and impossible differential fault analysis on aes," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 274–291, 2011.