Performance Evaluation of Protocols Resilient to Physical Attacks

Sylvain Guilley*†, Laurent Sauvage*†, Jean-Luc Danger*†, Nidhal Selmane* and Denis Real‡.

* Institut TELECOM / TELECOM ParisTech, CNRS LTCI (UMR 5141),
Département COMELEC, 46 rue Barrault, 75 634 PARIS Cedex 13, FRANCE.
† Secure-IC S.A.S., 37/39 rue Dareau, 75 014 PARIS, FRANCE.
‡ DGA/Information Superiority, La Roche Marguerite, 35 174 BRUZ, FRANCE.

***Abstract*—Cryptographic implementations are vulnerable to physical attacks. Many countermeasures to resist them have been proposed in the past. However, they are all specific to a given attacker and allow to mitigate the risk only up to a certain level: improved attacks on those countermeasures can most of the time be devised. Therefore, a new trend consists in making cryptographic implementations resilient to physical attacks. This strategy makes it possible to prove the countermeasure against all possible types of attackers captured by a security model. Several resilient schemes for the protection of block ciphers exist. For a given security objective, they all permit to reach the same security level. Therefore, they differentiate only according to their efficiency. We first show that the genuine versions of these protocols achieve different I/O bandwidth and computational performance. Our second contribution is to improve those protocols thanks to a message blinding, assuming passive attacks require more than two traces to be successful. Then, we bring as a third contribution the fact that the improved versions of the protocols are very much alike, and that the difference between them depends only from the specific details of their instantiation.**

I. Introduction

Implementation-level attacks aim at retrieving secrets concealed in embedded devices. They consist in passive attacks, where the attacker records some physical side-channel leaked out of the circuit, and active attacks, where the attacker perturbs the circuit in a view to have it output incorrect results. Unless the circuit features dedicated countermeasures, an attacker manages to exploit the leakage or the errors to successfully retrieve the key.

For a long time, engineers have attempted to make such a key retrieval harder, by essentially removing the dependence between the inner secret and the leakage by making the side-channel constant or random, and by detecting any fault injection. However, it has appeared that if those techniques manage to increase the attacks difficulty, they do not totally prevent them. Second order flaws have appeared and combined attacks also have shown up. Thus, assessing the exact security gain conveyed by those techniques has become hard.

For this reason, a recent trend has been to promote provable countermeasures, for which a rationale of attack impossibility can be demonstrated. The *resilience* strategy to thwart passive and active attacks meets this requirement. The advance of this technique is to allow a circuit to leak information and to output incorrect results, as long as they do not compromise the keys. In asymmetric cryptography, such schemes are already known [1]. Now, protecting symmetric cryptography remains a challenge, because block ciphers are less structured.

The rest of the article is structured as follows. The known resilient schemes are described in Sec. II. As those schemes were initially presented in different adversarial models, we mention in Sec. III the plurality of the existing risks and settle a common security objective. Within this shared security framework, the various protocols resilient to physical attacks differ only by their performances. To compare their relative performances, we consider in Sec. IV two scenarios that correspond to two typical use cases. Then, in Sec. V, we introduce a novel resilient protocol that takes the most of the session keys while remaining secure against both passive and active implementation-level attacks. It trades some cryptographic-grade primitives for lower cost primitives, thereby increasing the performances beyond the state-of-the-art without jeopardizing the security. Eventually, Sec. VI concludes the paper and opens some perspectives, notably on the need for implementation-level robust lightweight primitives, that can be instantiated advantageously in low cost but highly secure embedded devices, while maintaining the formality of the security analysis.

II. State-of-the-Art

Several resilient computation schemes have emerged recently. Most of them are "leakage-resilient", *i.e.* resilient against passive side-channel attacks, that consist merely in observing the physical emanations leaking from a cryptographic device. Some *ad hoc* constructs (*i.e.* not based on standardized algorithms, such as AES) for leakage-resilient stream ciphers [11] and signatures [3] have for instance been described.

However, many industrial applications require the cryptosystem to have its security based on standardized primitives. We therefore concentrate in the rest of this article on the protection of a block cipher g_k, such as the advanced encryption standard (AES), against physical attacks aiming at recovering its secret key k. To our best knowledge, only four publications tackle with protocol-level resilience for block ciphers. Three of them, namely *indexed key update* (abridged IKU and detailed in Sec. II-A), *fresh re-keying* (abridged FRK and detailed in

Sec. II-B) and *all-or-nothing encryption* (abridged AONE and detailed in Sec. II-D), deal with leakage resilience. The second protocol, FRK, is described for single block encryptions. As will be discussed in Sec. III, under this limitation, IKU and FRK can be proved resilient to active attacks. At the opposite, the fourth protocol, called fault injection resilience (abridged FIR and detailed in Sec. II-C), is only resilient against fault injection attacks, and is also described from single-block encryptions. The AONE protocol is also resilient against fault injection attacks, when multiple blocks are encrypted.

A. Indexed Key Update (IKU)

The leakage-resilience ensures that an attacker cannot retrieve the full secret key k, assuming two hypotheses. The first one is that only computations that involve the key induce leakage. The second is that one encryption leaks much less information than the whole key.

These hypotheses indeed reflect some true facts about practical side-channel analysis. The first one is a consequence of the way nowadays electronic circuits work. They are implemented in CMOS logic, that is leakage-free in static mode. Put differently, CMOS is built on purpose not to conduct any current if no net changes. At the opposite, when there is some activity, then the nets that toggle produce a current. The aggregation of those current make up the side-channel. The second hypothesis is a consequence of this fact: in a non-invasive setup, the attacker is not able to probe an node of the circuit. Instead, through the passivation layers, she can reasonably monitor the sum of the leakage of many nets in the vicinity of her sensor. Therefore her side-channel inevitably contains some algorithmic noise, *i.e.* random activity caused by the neighbour nets. If we also take into account the finite bandwidth of the sensor and the finite accuracy of the digitalizing apparatus, it appears clearly that several measurements will be necessary, simply to get rid of all this noise.

In this context, Paul C. Kocher suggests in [7, §4] to update the key on a regular basis. Typically, an evaluator can estimate the number of key manipulations after which the full key can be recovered by an attack, because sufficient information can be garnered to overcome the effect of the random noise. Now, this number depends on the characteristic of the acquistion campaign; thus, this paper [13] suggests to use a success rate metric to estimate the strength of an attack. Using this tool, it is possible to define a number η of encryption for which the success rate is below a given threshold, say 1%. The initial proposal of Paul C. Kocher is to hash the secret key every η encryptions, and to continue the encryptions with the result of the hash $k := h(k)$. Again, after η other key usages, the secret is replaced by its hash value. Because of the cryptographic properties of hash functions, the partial knowledge of the key before the key hash cannot be capitalized to break the new session key. The same noting can be done in the other way round: the partial information about the current key is of no help to deduce constructive information about the less iterated hashed keys. It is in this respect that this regular key update is resilient against passive attacks.

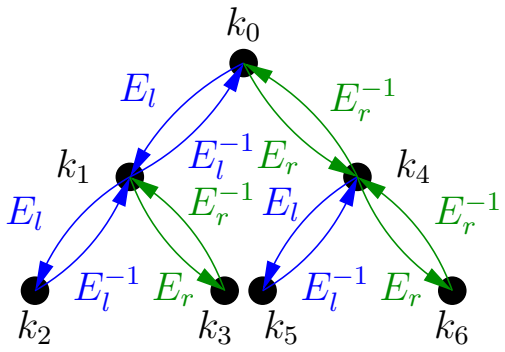

Fig. 1. Illustration of IKU in a binary tree for a depth $D = 3$.

However, this key update is not appropriate to the computation with multiple parties. For the sake of illustration, we assume that the party A is a vulnerable device (say a smartcard), that communicates with many correspondents B (supposed to be secure). Now, if A has already been hashing the key a large amount of times (say $1,000,000$ times), then the next B it interacts with needs to start computing $1,000,000$ hashes on the primary secret key k before being synchronized with A. This situation seems unrealistic, and will get worse as the number of used keys increases.

Therefore, Paul C. Kocher introduces in [6] a notion of session key tree. We assume binary trees, but similar constructs can be obtained for arities greater than two. The system assumes a finite number of keys, $2^D - 1$ for some integer D. For instance, in [6], D is chosen equal to 39. In the sequel, we simply note that it is a small constant, that satisfies $D = \mathcal{O}(1)$ when the number of blocks to encrypt (n) increases. The tree is rooted by k, and constructed recursively. The two sons of a node κ are respectively $E_l(\kappa)$ and $E_r(\kappa)$, the result of the encryption of κ by one of the two reversible functions E_l or E_r. Typically, $E_{\{l,r\}}$ are encryption functions, using two different keys that are made public. Indeed, the knowledge of the transformation of one session key into another does not convey any information to a prospective attacker if the nodes value is secret. In addition, publishing the encryption keys for $E_{\{l,r\}}$ reduces the amount of shared secrets to be otherwise safely concealed in every A and B. Now, the correspondents derive their successive session keys by a depth-first left-to-right tree traversal. Moving downwards left (*resp.* downwards right, upwards left, upwards right) is achieved by the application of E_l (*resp.* E_r, E_l^{-1}, E_r^{-1}). The current key is indexed by its order $C \in [\![0, 2^D[\![$ in the tree traversal. We denote k_C the corresponding key, and note $k_0 = k$, because, by convention, the root of the tree has index $C = 0$. The figure 1 illustrates this key indexation mechanism.

The session key is agreed on between A and B by first comparing their counters C. They select the greatest of the two C for the shared index. Then, there exists an algorithm to reach any node from any other node using between 0 and $2D-2$ calls to either of $E_{\{l,r\}}^{\{+1,-1\}}$. Once this first session key

is fetched, the subsequent keys $k_{\{C+1,C+2,\cdots\}}$ are retrieved each thanks to only one call to either $E_{\{l,r\}}^{\{+1,-1\}}$, because a single displacement is required since consecutive keys are direct neighbours in the tree.

B. Fresh Re-Keying (FRK)

The previous IKU scheme has several drawbacks:

- the number of possible keys is limited to $2^D - 1$ although the key space is must larger;
- the key agreement requires several calls to encryption/decryption functions, and this number of calls depends on the key index C;
- the correspondent A must have some tamper-proof non-volatile memory (NVM) to store C.

The paper [10] introduces an interactive session key derivation algorithm that addresses all the shortcomings of IKU. For this purpose, the authors of [10] replace the key search in a tree by a random key generation, thanks to a randomized bijection noted f. In practice, this function takes in input the root key k and a random number r, and outputs a fresh session key $k^* = f(k, r)$. A sends to B the random number r so that B is able to reproduce the cryptographic computations done by A. It is straightforward to understand that this fresh re-keying protocol can generate all possible keys, in one go, and in a state-less manner.

Nevertheless, one immediate shortcoming of this FRK arises from its interactivity. In the long run, this algorithm is more I/O consuming, since A must send a random number along with each block of encrypted data, whereas in IKU, the synchronization with C is done once at the beginning, and subsequently A and B remain in phase if they implicitly know when to increment C.

Despite of this consideration, it is worth noting that FRK can be instantiated in an implementation that enjoys a remarkably efficient session key derivation algorithm. The authors of [10] indeed underline that the security features of the protocol can be partitioned in two independent problems. On the one hand, the block cipher g keyed with $k^* = f(k, r)$ is responsible for the scheme's robustness against cryptanalytic attacks. On the other hand, the resilience against physical attacks is confined in f. There are two reasons for that. First of all, if the block cipher is invoked fewer times than the number of times η where a side-channel attack have fair chance to succeed, then k^* will definitely remain out of reach of an attacker. Now, not knowing k^*, the attacker cannot inverse $f(\cdot, r)$ if it has a good diffusion property. Second, another attack path would be to examine the leakage of f: indeed, this function is fed by an unknown secret k mixed with a known varying input r. This setup is canonical for a side-channel attack. Nonetheless, as defended in [10], f can be chosen to both fulfill the diffusion property and to be easily protected against side-channel attacks. In the example discussed in [10], the very nature of f (the multiplication $*$ in a given ring) makes it easily protected by masking. Also, $f = *$ can be implemented much more efficiently than a complete cryptographic bijection (*e.g.* $f =$ AES).

C. Fault Injection Resilience (FIR)

Avoiding faults in a circuit is a difficult task, since whatever detection scheme, it is always possible, by chance, to substitute a valid data by an other valid one, that will obviously not be detected. On the contrary, the resilience approach against fault injection attacks consists in tolerating errors, but also in denying the attacker from exploiting them. We call fault injection resilient (FIR) a scheme where the attacker can neither *choose* nor *influence* the input of the encryption algorithm. This way, intuitively, the attacker has no means even to know if the result is faulted or not, and even less to know how the ciphertext is faulted. In particular, differential fault attacks are impossible since it is impossible to collect twice the ciphertext corresponding to the same plaintext, and safe errors are equally impossible since the attacker cannot distinguish between the correct and the purportedly faulted ciphertext. This FIR model would certainly deserve a more formal definition. Nonetheless, we continue with this FIR notion, defined as the input *non-forgeability* and *non-malleability*.

An example of FIR is given in [5, §3-A]. The proposed way to avoid an attacker from choosing or influencing the plaintext is to blind it with a random number r of the same size. Thus, instead of returning the encryption $g_k(x)$ of plaintext x, the algorithm returns the couple $(g_k(x \oplus r), r)$. This way of proceeding doubles the requirement for the I/O bandwidth, but is otherwise computationally equivalent to the plain unprotected g_k, if we neglect the XOR operation involved in the blinding compared to an encryption g_k.

D. All-Or-Nothing Encryption (AONE)

The AONE [8], [9] builds its security by denying the attacker from knowing the plaintext and the ciphertext. However, in AONE, the key is not updated: it remains k throughout the encryptions. The suggested construct is an all-or-nothing transform (AONT [12]), applied *full* on the plaintext and *partial* on the ciphertext.

We remark here that the AONT is too strong a preprocessing on the plaintext; indeed, the goal is simply to make it unpredictable to the attacker. Thus, a simple *partial* AONT, such as a lightweight mask generation function (MGF) described in Fig. 2, do achieve a randomization of the plaintext. The MGF consists in the iterated application on a random number r of a deterministic function f that can be abstracted as a random oracle (*e.g.* a lightweight encryption function with a public key or a lightweight hash function). This MGF adds one extra block r to be encrypted; this block is drawn randomly by A, and discovered upon decryption by B.

For AONE to be effective as such against passive attacks, the ciphertext shall also be blinded. However, this requires an initial secret exchange, which is a serious drawback of the approach. Indeed, sharing a secret requires an operation such as a Diffie-Hellman (DH) exchange [2]. Now, low cost devices that rely on symmetric cryptography are not equipped with the hardware to conduct DH. In addition, the DH exchange is very expensive in terms of time and code size. Therefore, this pass is really deterrent for the adoption of the AONE.

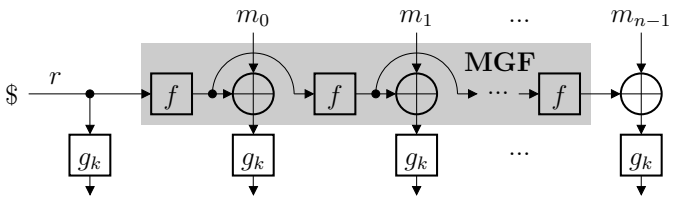

Fig. 2. Resilient MGF, used as partial AONT.

E. Synthesis about the State-of-the-Art

In the sequel, we discard FIR as such, because it does not protect against passive side-channel attacks. AONE, even in our optimized version, is also discarded because of the unrealistic requirement for an initial secret sharing in addition to the key. Nonetheless, the principle of input non-forgeability and non-malleability of FIR and of our optimized version of AONE will be reused latter on in Sec. V to armor IKU and FRK against active attacks.

III. Security Model and Security Target

The goal of this section is twofold. First of all, we define the threats that shall be taken into account by the resilience schemes we intent to compare, and also quantity them. Then, we set up the security properties we want our resilient schemes to meet.

A. Formalization of the Risks

Passive and active attacks are equally likely to be applied on an embedded system. A reasonable attacker will certainly choose the one that is the easiest.

Regarding passive attacks, we adopt the same classifications of attacks as presented in [10]. A primitive that leaks the whole secret is said attackable by the simple power attack (SPA). Now, with such a primitive, it is very hard if not impossible to compute securely. Thus, we define primitives to be SPA-resistant if they do not disclose all the secret key by only one observation by attacker. The differential power analysis (DPA) is a statistical attack that exploits the dependence of the measurements in a binary sensitive variable. We reuse the notation η introduced previously to quantify the amount of measurements for which not enough bits of the secret have leaked to compromise it. We thus assume that an SPA-resistant primitive can be safely called η times. As we wish to build a resilient protocol without resorting to expensive *ad hoc* countermeasures, we will only consider off-the-shelf primitives. Now, any respectable intellectual property (IP) cipher can be expected to be SPA-resistant but not DPA-resistant. Thus, in the sequel, we assume that g_κ is not called more than η times with the same key κ.

Active faults can be extremely effective. When g is the AES-128, then as few as one fault can be enough to extract the 128-bit key: this has notably been shown in this differential fault analysis (DFA) [14]. As this attack is differential, it actually requires the knowledge of one correct and one (specially crafted) faulted. Thus, unless special care is taken (such as explained in the improved AONE in Sec. II-D), it shall not be considered secure to let the protocol encrypt two blocks with the same key. Additionally, we note that $\eta \gg 2$, *i.e.* resisting to passive leakage is easier than resisting to fault injection attacks, in terms of number of queries.

B. Common Set of Security Objectives

We consider the case of equipments sharing a common secret k and willing to use it in a block cipher, either for the purpose of authentication or (non-exclusively) for the purpose of data encryption. As already mentioned, we demand that no other secret be shared by the devices. The security of the block cipher encryptions must rely only on the resilient usage manipulation of k.

Our security objective is to compare resilience schemes that do not disclose the secret k, for a number of large transactions (much greater than η but all the same smaller than $2^D - 1$, for IKU to be admitted in the competition), and in the context of passive and active attacks. This must be achieved using only an off-the-shelf cryptographic primitives for g, attackable as such (*i.e.* without resilience) with η traces passively and with 2 encryptions in active attacks.

The functionality is the encryption, that must be implemented by a function g that is cryptographically strong. Of course, the other functions involved in the implementation of the resilience can be *ad hoc*, and can be only resistant against active and passive attacks. Such primitives might be much less costly than cryptographic primitive. Their use can improve the efficiency (cost in terms of speed and required hardware/software) of the resilient protocol.

So far, two protocols fulfill the security requirements: IKU with single block encryptions (otherwise DFA becomes possible) and genuine FRK (with a single block also). They are sketched in Tab. I for one block encryptions. In the DFA-proof setup, this protocol is merely repeated n times to request for n encryptions.

IV. Performance Assessment

A. Authentication and Files Encryption

We consider two scenarios: symmetric authentication of devices and large files encryptions. The first case is a mere challenge-response involving the encryption of one block, whereas the second one depicts more the case of an electronic passport encrypting an identity facial picture (4 kBytes make up 512 triple-DES blocks or 256 AES-$\{$128,192,256$\}$ blocks).

B. Performance Figures

The protocols can be appreciated according to various criteria, depending on which resource is the most limiting on the targeted device. First of all, the presence of NVM can be considered an option in extremely price-constrained devices. Or also, when migrating one application on a device with just enough NVM to accommodate the secret key k from a low protection level to a resilience-protection type, it is not an alternative to consider more NVM since the upgrade shall be done at constant resources (which is also why we do not

TABLE I
PASSIVE AND ACTIVE RESILIENT ENCRYPTION BY A OF A MESSAGE SENT BY B.

Step	Single-block IKU		
#1	A sends C_{A} A receives C_{B}	$\longrightarrow$ $\longleftarrow$	B receives C_{A} B sends C_{B}
#2	A computes k_C as $k_C = k_{\max(C_{\mathsf{A}},C_{\mathsf{B}})}$		B computes k_C as $k_C = k_{\max(C_{\mathsf{A}},C_{\mathsf{B}})}$
#3	A receives m	$\longleftarrow$	B sends m
#4	A computes $y = g_{k_C}(m)$		
#5	A sends y	$\longrightarrow$	B receives y

Step	Single-block FRK		
#1	A sends r	$\longrightarrow$	B receives r
#2	A computes k^* as $k^* = f(r,k)$		B computes k^* as $k^* = f(r,k)$
#3	A receives m	$\longleftarrow$	B sends m
#4	A computes $y = g_{k^*}(m)$		
#5	A sends y	$\longrightarrow$	B receives y

TABLE II
SUMMARY OF THE PERFORMANCES OF VARIOUS RESILIENT PROTOCOLS (FOR 1 TO n BLOCKS).

Protocol	I/O [bit]	Performance
1-bl. IKU	$2D + 2B$	$(2D-2)[E] + [g]$
1-bl. FRK	$3B$	$[f] + [g]$
n-bl. IKU	$2D + 2Bn$	$(2D-3+n)[E] + n[g]$
n-bl. FRK	$3Bn$	$n \cdot ([f] + [g])$
n-bl. IKU+	$2D+(1+\frac{\eta}{\eta-1})nB$	$(2D-3+\frac{n}{\eta-1})[E]+n(\frac{\eta}{\eta-1}[g]+[f])$
n-bl. FRK+	$(\frac{2\eta}{\eta-1})nB$	$\frac{n}{\eta-1}[f] + n(\frac{\eta}{\eta-1}[g] + [f])$
n-bl. IKU*	$2D + 2Bn$	$(2D-3+n)[f] + n[g]$
n-bl. IKU+*	$2D+(1+\frac{\eta}{\eta-1})nB$	$(2D-3+\frac{n}{\eta-1})[f]+n(\frac{\eta}{\eta-1}[g]+[f])$
n-bl. FRK+H	$B + (1 + \frac{\eta}{\eta-1})nB$	$\frac{n}{\eta-1}[f] + n(\frac{\eta}{\eta-1}[g] + [f])$

- IKU & IKU* require ..... NVM but no TRNG;
- IKU+ & IKU+* require ..... both NVM and TRNG;
- FRK, FRK+ & FRK+H require ..... TRNG but no NVM.

investigate [4]). Second, protocols can be classified according to their requirement for a true random number generator (TRNG). Then, the number of exchanged messages can also be limiting, especially for contact-less devices. Eventually, the complexity in terms of execution time and processing power is a third parameter to take into account. In this respect, the cryptographic primitives are always considered most costly than *ad hoc* lightweight primitives that fulfill only one requirement, such as diffusion, while remaining at least SPA-resistant. Indeed, SPA-resistance is assumed to be a prerequisite for all the blocks involved in the execution of the protocol.

C. Results for State-of-the-Art Protocols

The IKU requires some NVM, *a minima* in a quantity equal to the key tree depth (D bits) to store the current position C. On the contrary, FRK works without NVM. Symmetrically, IKU is deterministic, whereas FRK demands a TRNG. Regarding the amount of data to be sent in an authentication session between A and B, the key establishment procedures (step #1 in Tab. I) require $D + D$ bits in IKU and B bits (where B is the cipher g block size) for FRK. Taken into account the one-block reception (step #3) and reemission after encryption (step #5), one ends up with $2D + 2B$ for IKU and $3B$ for FRK. If we note $[X]$ the performance of operation X, then IKU costs between $1 \times [E]$ to $(2D-2) \times [E]$ to compute the session key on the tree, where E is one of $E_{\{l,r\}}^{\{+1,-1\}}$. Let us consider worst cases. If we neglect small comparisons and focus on operations on data and keys, then IKU costs in total $(2D-2) \times [E] + [g]$ and FRK $[f] + [g]$.

Now, if instead of the authentication case, we focus on the large file encryption case, then n applications of g shall be considered. In IKU, the exchange of the positions in the tree is done only for the first block; afterwards, we can assume A and B implicitly know that the next key can be found at the next position, with one application of $E_{\{l,r\}}^{\{+1,-1\}}$. As far as FRK is concerned, encrypting multiple blocks consists in replaying the same single block protocol with a new random r each time. Thus, for long messages ($n \gg 1$), IKU becomes more bandwidth-efficient than FRK, whereas FRK keeps its performance advantage (since $[f] < [E] = [g]$, because f is lightweight whereas E and g are full-fledged cryptographic block ciphers). These results are summarized in the four first rows of Tab. II.

V. IMPROVEMENT OF THE STATE-OF-THE-ART IN THE ENCRYPTION OF LARGE FILES SCENARIO

A. Armoring IKU and FRK on $n > 1$ Blocks against Fault Attacks: IKU+ and FRK+

We present IKU+ and FRK+, that are multi-block versions of IKU and FRK. The two latter protocols could not use multiple blocks because of fault attacks. Now, if a MGF is applied on the plaintext, then the plaintext actually becomes non-forgeable and non-malleable, thus fault injection resilient. Therefore, one session key can be reused for up to η blocks — beyond, DPA is a concern.

To simplify the comparisons, we assume that n is large ($n > \eta$), and multiple of η. With one session key, we can safely encrypt η blocks. Now, in η blocks, one is reserved to encrypt the random nonce r involved in MGF (recall Fig. 2). Thus, $\frac{n}{\eta-1}$ session keys are required. In IKU+, the first key look-up still costs $(2D-2) \cdot [E]$, and the other keys continue to cost $[E]$ each, however their number is reduced from $n-1$ down to $\frac{n}{\eta-1} - 1$. The introduction of the MGF adds $\eta - 1$ operations "f" per session key, hence n calls to f. However, the bandwidth is unchanged. FRK+ profits from the MGF both in terms of bandwidth and performance, as shown in Tab. II.

B. Improving IKU with Lightweight Key-Update: IKU+*

FRK makes use of a lightweight primitive f instead of a cryptographic-grade primitive (E) to derive the session keys.

The same could be done for IKU. We call IKU* (*resp*. IKU+*) the version of IKU (*resp*. IKU+) where E is replaced by a lightweight ersatz (of similar complexity as the f in [10]) that does not alter the security level since they manipulate unknown quantities.

Related keys attacks have been reported recently on AES-192 and AES-256. They are cryptanalytic attacks that require two or more encryptions with related keys, *i.e.* differing by only a few bits. To be immune from these attacks, one must make sure that the update function is not two trivial. However, a primitive such as $f = *$ certainly has an high enough dispersion to prevent such related keys to be produced frequently.

C. Synchronous Session Keys Update by Iterative Hashing: FRK+H

The first session key must be agreed on, deterministically as for IKU and probabilistically as for FRK. But the next session keys can be obtained from another protocol, such as iterative hashing of the first session key (as suggested in [7, §4]). For FRK+, this allows to still improve on the I/O bandwidth, without altering the resilience property. This scheme is called FRK+H, and its performance is given in Tab. II. For IKU+*, this would trade a lightweight key update with f on the tree with a lightweight hash, we assume to have the same cost.

At this stage, we have optimized as much as possible IKU and FRK: the best protocols for large files encryption are IKU+* and FRK+H. The result is that

- in term of I/O bandwidth, the requirements are the same when $n \to \infty$ (namely $(1 + \frac{\eta}{\eta-1})nB$, up to a negligible constant);
- Computation-wise, both require exactly $n\frac{\eta}{\eta-1}$ calls to g and about $n(1 + \frac{1}{\eta-1})$ calls to a lightweight f.

Thus, the differences observed for short messages (authentication case) tend to fade and asymptotically, the optimized protocols are equivalent when dealing with the encryption of large messages.

D. Other Implementation-Dependant Considerations to Tune the Resilience Schemes

Each time the protocol reduces the number of key updates, as in Sec. V-A, the key schedule step of g is saved. Now, this step is both timing consuming (especially on AES) and source of an extra leakage.

VI. Conclusions and Perspectives

We have investigated resilient computation schemes for both hardware and software implementations. Our study shows that, amongst the known schemes, those based on a regular key update are effective. Two solutions (IKU and FRK) exist, depending whether the key sequence is deterministic or probabilistic. We show that using state-of-the-art protocols, FRK is always more efficient for single-block encryption, whereas IKU is always less I/O consuming for large files encryptions. However, as such, IKU and FRK can only use one-block payload exchange, because of fault injection analyses. By blinding the input, we show that multi-blocks can be used, which improve the performance (leading to the new protocols we nickname IKU+ and FRK+). In addition, we propose another series of improvements, after which the two schemes tend to be equivalent (when the number of blocks to process n becomes larger and larger). Thus, we confirm that, for an identical security objective, the use of lightweight primitives in conjunction with cryptographic primitives can indeed enhance the efficiency of the protocol. As a perspective, we expect interesting researches on this topic to continue make the cost of resilience-based protections more acceptable.

References

[1] J.-S. Coron and A. Mandal. PSS Is Secure against Random Fault Attacks. In *ASIACRYPT*, volume 5912 of *LNCS*, pages 653–666. Springer, December 6-10 2009. Tōkyō, Japan.

[2] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Trans. on Info. Theory*, 22(6):644–654, 1976.

[3] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-Resilient Signatures. In *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360. Springer, February 9-11 2010. Zurich, Switzerland.

[4] J. Guajardo and B. Mennink. On Side-Channel Resistant Block Cipher Usage. In M. Burmester, G. Tsudik, S. S. Magliveras, and I. Ilic, editors, *ISC*, volume 6531 of *LNCS*, pages 254–268. Springer, 2010.

[5] S. Guilley, L. Sauvage, J.-L. Danger, and N. Selmane. Fault Injection Resilience. In *FDTC*, pages 51–65. IEEE Computer Society, August 21 2010. Santa Barbara, CA, USA.

[6] P. C. Kocher. Leak-resistant cryptographic indexed key update, March 25 2003. United States Patent 6,539,092 filed on July 2nd, 1999 at San Francisco, CA, USA.

[7] P. C. Kocher. Design and Validation Strategies for Obtaining Assurance in Countermeasures to Power Analysis and Related Attacks, September 26-29 2005. Honolulu, Hawai, USA; NIST's Physical Security Testing Workshop. Website: http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/physecdoc.html.

[8] R. P. McEvoy, M. Tunstall, C. Whelan, C. C. Murphy, and W. P. Marnane. A differential side-channel analysis countermeasure. European Patent Application (EP 2148462 A1), filled in 27.01.2010.

[9] R. P. McEvoy, M. Tunstall, C. Whelan, C. C. Murphy, and W. P. Marnane. All-or-Nothing Transforms as a Countermeasure to Differential Side-Channel Analysis. Cryptology ePrint Archive, Report 2009/185, April 30 2009. http://eprint.iacr.org/2009/185.

[10] M. Medwed, F.-X. Standaert, J. Großschädl, and F. Regazzoni. Fresh Re-Keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices. In *AFRICACRYPT*, volume 6055 of *LNCS*, pages 279–296. Springer, May 03-06 2010. Stellenbosch, South Africa.

[11] K. Pietrzak. A Leakage-Resilient Mode of Operation. In *EUROCRYPT*, volume 5479 of *LNCS*, pages 462–482. Springer, April 26-30 2009. Cologne, Germany.

[12] R. L. Rivest. All-or-Nothing Encryption and the Package Transform. In *FSE*, volume 1267 of *LNCS*, pages 210–218. Springer, January 20-22 1997. Haifa, Israel.

[13] F.-X. Standaert, T. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *EUROCRYPT*, volume 5479 of *LNCS*, pages 443–461. Springer, April 26-30 2009. Cologne, Germany.

[14] M. Tunstall and D. Mukhopadhyay. Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault. Report 2009/575, 2009. http://eprint.iacr.org/2009/575, to appear in WISTP 2011 (Springer LNCS 6633, Heraklion, Greece).