On the Bright Side of Darkness: Side-Channel Based Authentication Protocol Against Relay Attacks

Guillaume Dabosville[1], Houssem Maghrebi[2], Alexis Lhuillery[3], Julien Bringer[4], and Thanh-Ha Le[5]

[1] Groupement des Cartes Bancaires (GIE CB)
guillaume.dabosville@gmail.com
[2] Underwriters Laboratories (UL)
houssem.mag@gmail.com
[3] Sorbonne Université
alexis.lhuillery@orange.fr
[4] Smart Valor
julien.bringer@gmail.com
[5] Trusted Labs
lethanhha.work@gmail.com

Abstract. Relay attacks are nowadays well known and most designers of secure authentication protocols are aware of them. At present, the main methods to prevent these attacks are based on the so-called distance bounding technique which consists in measuring the round-trip time of the exchanged authentication messages between the prover and the verifier to estimate an upper bound on the distance between these entities. Based on this bound, the verifier checks if the prover is sufficiently close by to rule out an unauthorized entity. Recently, a new work has proposed an authentication protocol that surprisingly uses the side-channel leakage to prevent relay attacks. In this paper, we exhibit some practical and security issues of this protocol and provide a new one that fixes all of them. Then, we argue the resistance of our proposal against both side-channel and relay attacks under some realistic assumptions. Our experimental results show the efficiency of our protocol in terms of false acceptance and false rejection rates.

Keywords: authentication protocol, relay attacks, side-channel attacks, physical leakage.

1 Introduction

Relay Attacks VS Authentication Protocols. A *relay attack* is a form of *man-in-the-middle* attack where the attacker merely relays the verbatim messages between a verifier and a prover, to authenticate to the verifier as a legitimate prover. In such a context, the attacker is usually close to the verifier and

claims being the legitimate prover while the latter is not in the neighborhood. Such an attack is particularly well-suited to get around access controls, *e.g.* to get inside a secure location or to unlock the doors of a vehicle. Without loss of generality, we shall consider in the sequel that the prover is a contactless Secure Element (SE) and the verifier is a contactless reader.

The current trend to include Near Field Communication (NFC) (ISO/IEC 14443) technology into mobile phones significantly simplifies skimming and relay attacks. Although the NFC protocol requires that the prover and the verifier are in close proximity, this constraint does not help to counteract relay attacks. Indeed, in [**?**] it is shown how to set up a relay attack by placing a *proxy-token* in the range of a contactless reader to relay queries from the verifier to a *mole* which forwards to the genuine prover the requests from the genuine verifier. The mole also sends back the responses from the genuine prover to the proxy-token which delivers them to the genuine verifier. Most recently the authors in [**?**,**?**] have shown how smartphones equipped with an NFC antenna can be used efficiently as a generic relay attack platform.

To thwart relay attacks, Brands and Chaum introduced in 1993 the so-called *distance bounding protocols* [**?**].

The core idea of these protocols is to measure the round-trip time of the authentication messages exchanged between the prover and the verifier. Then, based on this propagation time, the verifier decides whether the prover is within the distance and hence discriminates a legitimate prover from an attacker. Although the idea has been introduced many years ago, it is only quite recently that distance bounding protocols have been investigated and several designs have been proposed in the literature [**?**,**?**].

On the Bright Side of Side-Channel Leakage. Before detailing our proposal, we provide in this section a survey of some recent research lines suggesting the use of side-channel information constructively to enhance, or as an alternative to, existing cryptographic protocols.

To avoid counterfeiting of Integrated Circuits (ICs), authors in [**?**] have proposed a watermarking based technique. It consists in inserting a software manipulating a sensitive variable which simply computes the Exclusive-OR (XOR) operation between an internal 8-bit counter and an 8-bit secret key and then applies the Sbox of the AES to the result of the XOR. The internal counter increases its value on each run of the software. To check whether a device is genuine or not, the issuer of the device provides a reference device containing the same software as in the device under test (if the latter is not a counterfeit). By measuring several power traces from both devices and then by computing the degree of correlation between their respective leakages, the issuer can conclude that the audited device is a legal copy if the correlation is sufficiently high (*i.e.* greater than a fixed threshold). Unfortunately, the authors only use an 8-bit length key. Moreover, each time it is required to check the authenticity of a device, the internal 8-bit counter is reset on both devices (the device under test

as well as the reference device), making the sequence predictable and subject to side-channel attacks[6].

In another work, Kerckhof *et al.* in [**?**] have used some techniques from side-channel analysis still in the context of IP protection. The typical use case on which they have focused on is the implementation of some customized ciphering algorithms. An estimation of the coefficient of correlation is again used but now (despite of the context of symmetric ciphers), no sensitive variable is involved in the computation of the correlation. The authors have rather used the correlation as a tool to correlate features extracted from an acquired signal (which represents the execution of a specific implementation without paying attention to the manipulated data) with features extracted from a reference device embedding the same IP as in the device under test. Several other works have suggested to use side channel for IC fingerprinting or Trojan detection [**?**,**?**].

To avoid relay attacks, authors in [**?**] and [**?**] have proposed authentication protocols based on the use of physical leakage. In [**?**], the authors simply suggested the use of a dedicated binary xor instruction which, depending on the resulting bit, leaks exaggeratedly (*e.g.* if the resulting bit is 1) or not (*e.g.* if the resulting bit is 0). Such a technique allows the authors to only use a so-called simple side-channel analysis. In [**?**], the prover performs on his side some cryptographic computations while the verifier acquires the corresponding physical leakage (*e.g.* the electromagnetic radiation). Assuming that the verifier and the prover share a symmetric key K, the verifier analyses the leakage generated by a nearby SE and authenticates it "physically" if the correlation between the acquired leakage and the associated predictions made by the verifier is sufficiently high. Such an authentication associating classical authentication techniques with statistical side-channel analysis is called *side-channel authentication*. According to us, this new paradigm is really interesting since it may indeed allow to detect relay attacks in the field. The fact that it is required to acquire the electromagnetic signal of the prover (which is is typically a smartcard) may not be unrealistic depending on the considered use case. We acknowledge that adding the means for signal acquisition in a so-called payment terminal is probably not an option. This situation is not so clear when talking about relay attacks on car immobilizers (where the key is more or less a smartcard) or when talking about control access to a public transport network. In both cases it would surely be possible to embed an equipment to acquire the leakage of the prover: inside the driver's door of the car and inside the entrance gate to the public transport network. Of course this countermeasure comes at a price and its implementation will depend on how relay attacks impact the overall fraud.

Our Contribution. In this work, we propose a new authentication protocol that (1) follows the side-channel based approach suggested in [**?**] and [**?**] and (2) fixes its practical and security issues that we detail hereafter. According to our understanding, in [**?**] the authors propose four schemes in which both

[6] If the reset value of the internal counter is unknown to the attacker, she must recover it, which is far from being an issue since it is only an 8-bit value.

parties, the prover and the verifier, share a symmetric key K that is used to perform AES computations (on both sides) using this secret key. The inputs of the AES computations are specified differently depending on the design that is considered. The main drawback of their proposal is that a customized N-round AES is required ($N = 1.000$ rounds in their experiments).

From a design point of view, this design constraint raises an issue because in the context of secure element only standardized cryptographic primitives should be considered (*i.e.* AES-128, AES-192 and AES-256). From a security perspective, the protocol proposed in [**?**] has some flaws that could be exploited by performing some side-channel attacks. The proposed N-round AES implementations are not protected against side-channel attacks. So, an adversary who can recover the physical leakage, is able to perform a statistical attack to recover the shared master key K. The reason behind not using the well-known side-channel countermeasures (*e.g.* masking, secret sharing) is that the leaked information is no longer exploitable and so the physical authentication cannot succeed.

Finally, from a practical perspective, assuming that the leakage model is uniform over the whole intermediate values of the N-round customized AES [**?**] is unsound in real hardware due to small load imbalances, process variations, routing, *etc.* For instance, authors in [**?**] have characterized using a stochastic approach the leakage of four successive AES Sbox outputs. The obtained results prove that the leakage is very unbalanced for each Sbox. Hence, in practice the authentication may fail since the correlation between the acquired leakage and the associated predictions may be too low when following such assumption on the leakage model.

To overcome these issues and to resist relay attacks, we propose a new authentication protocol that first ensures that only a genuine verifier can exploit the leakage from a prover, *i.e.* no attacker should be able to mount an attack based on side-channel analysis to recover the secret key K. Second, this protocol only uses standard cryptographic primitives (*i.e.* AES-128, AES-192 or AES-256).

Throughout several practical experiments (see Sec. 4), we argue that our proposal is secure against both side-channel attacks and relay attacks under some realistic assumptions.

Paper Outline. The paper is organized as follows. In Sec. 2, we briefly describe the AES block cipher and provide some useful notations. Then, in Sec. 3 we describe our new authentication protocol. To assess the efficiency of our proposal, a practical security evaluation is conducted in Sec. 4. Finally, Sec. 5 draws general conclusions and opens some perspectives for future work.

2 Background

2.1 AES

The Advanced Encryption Standard (AES) [**?**] is a block cipher that processes data blocks of 128 bits length and a variable secret key length (128, 192 or 256

bits). Hence, as specified by the standard [**?**], three different block-ciphers can be used: AES-128, AES-192, AES-256. Depending on the length of the key, the AES performs N_r rounds, with $N_r \in [10, 12, 14]$. The AES manipulates, all along its execution, an internal state which is a (4×4) matrix of bytes.

Algorithm 1 The Advanced Encryption Standard (AES)

Require: In: the 128-bit input, N_r: the number of rounds, $(k_i)_{0 \leq i \leq N_r}$: the round keys
Ensure: Out: the 128-bit output

$M_{-1} = In$
for $i = 0$ **to** $N_r - 2$ **do**
    $X_i = \text{AddRoundKey}(M_{i-1}, k_i)$
    $Y_i = \text{SubBytes}(X_i)$
    $S_i = \text{ShiftRows}(Y_i)$
    $M_i = \text{MixColumns}(S_i)$
end for
$X_{N_r-1} = \text{AddRoundKey}(M_{N_r-2}, k_{N_r-1})$
$Y_{N_r-1} = \text{SubBytes}(X_{N_r-1})$
$S_{N_r-1} = \text{ShiftRows}(Y_{N_r-1})$
$Out = X_{N_r} = \text{AddRoundKey}(S_{N_r-1}, k_{N_r})$
return Out

The AES involves four main operations: AddRoundKey, SubBytes, ShiftRows and MixColumns. As shown in Algorithm 1, at the i^{th} round these operations yield the so-called intermediate states of the AES and are denoted resp. by X_i, Y_i, S_i and M_i. A byte from an intermediate state, say X_i, is denoted by $X_i[l, c]$ with $(l, c) \in [0, 3]^2$ in the sequel.

2.2 Leakage Model

Let V be a sensitive byte (*e.g.* the variable $Y_i[l, c]$ of the i^{th} AES round), then it is often assumed that the leakage function $L(V)$ is well modeled by:

$$L(V) = \alpha_t \cdot \text{HW}(V) + \beta_t + W \quad \text{, where:} \tag{1}$$

1. HW(.) is the Hamming Weight function.
2. W is a Gaussian noise $\mathcal{N}(0, \sigma)$ with null mean and standard deviation σ.
3. (α_t, β_t) are some weighting values specific to each targeted sensitive byte of index (l, c) of the i^{th} AES round (*i.e.* $t = (l, c, i)$).

It is worthy to note that, in some specific cases, the real leakage function can be slightly different from the model we considered in (1). In such a case, one have to characterize the real leakage function by applying a *stochastic approach* as suggested in [**?**]. For the sake of simplicity, we assume that leakage function follows the model described in (1). In the meantime, we stress the fact that our proposal works well with any leakage function.

3 Protocol Proposal

As mentioned in the introduction, the protocol must ensure that only a genuine verifier can exploit the leakage from a prover, while avoiding any attacker attempting to mount a side-channel analysis. It must also consider some practical constraints such as the use of standard cryptographic APIs (*i.e.* avoiding customized AES implementations as required in [**?**] for instance).

3.1 Adversarial Model

In the sequel, we shall consider the following assumptions.

Assumption 1 (Attacker's profile) *We consider an attacker who can perform* (1) *side-channel attacks by capturing the side-channel leakage of the prover to try retrieving the secret key and* (2) *relay attacks to try authenticating to a verifier as a genuine prover.*

Assumption 2 (Impracticability of reproducing the physical leakage) *We assume that it is significantly difficult in practice for an attacker performing a relay attack to generate a copy of the side-channel information leaked by the genuine prover. This assumption is quite realistic and is merely justified in Sec. 3.3.*

Assumption 2 relies on the fact that even if an attacker succeeds in capturing, copying and replaying the side-channel leakage, then this procedure will take some time denoted Δ. Thus, replaying the side-channel information is equivalent to add some desynchronization in the trace which is well known to reduce the degree of correlation [**?**].

3.2 Protocol Specification

Overall Description. We provide in Fig. 1 an overview of our protocol. First, we assume that the master key K has already been shared between the verifier and the prover (*e.g.* either loaded during the personalization phase of the prover or by using the classical Diffie-Hellman key exchange protocol with certified public keys on both sides). The verifier initiates the protocol by sending R_V (a 64-bit random value) to the prover who replies by sending R_P (a 64-bit random value). Both then use the concatenation of R_V and R_P (denoted $R_V||R_P$) as an input to a secure *Key Derivation Function*, denoted KDF in Fig. 1, to generate $2N$ session keys $(K_0^{(0)}, K_1^{(0)}, \cdots, K_{2N-1}^{(0)}) = (K_0, K_1, \cdots, K_{2N-1})$ using the shared master key K. This secure KDF (*i.e.* orange box in Fig. 1) must be protected against classical side-channel attacks by implementing for instance some well-known masking countermeasures (see *e.g.* [**?**,**?**,**?**]). In addition, the verifier generates $q \times 2N$ extra random keys denoted $(K_0^{(i)}, K_1^{(i)}, \ldots, K_{2N-1}^{(i)})$ for $i \in [1, q]$. Now, to fix the practical issues of the protocol proposed in [**?**], we use a set of N AES encryption (*i.e.* the blue boxes in Fig. 1) instead of a customized

N-round AES. These AES executions, denoted *leaky AES* in the sequel, do not implement any countermeasure against side-channel analysis and are used to encipher the last N session keys

$(K_N, \cdots, K_{2N-1})$ using the first N ones $(K_0, K_1, \cdots, K_{N-1})$ as encryption keys.
On the prover side, the i^{th} AES encryption provides $C_i = \text{AES}_{K_i}(K_{i+N})$ and leaks all intermediate sensitive variables which are measured by the verifier.

The verifier on its side, computes $q+1$ such sequences of N AES and stores all relevant intermediate sensitive variables. The j^{th} output of the i^{th} sequence of N AES is denoted $C_j^{(i)} = \text{AES}_{K_j^{(i)}}(K_{j+N}^{(i)})$ for j in $[0, N-1]$ and i in $[0, q]$. Then, for each i^{th} executed sequence of N AES, the verifier computes a likelihood value $\mathcal{L}_i$ between the corresponding stored intermediate values and the acquired measurement while the prover was executing its N leaky AES.

At the end of this step, the verifier holds $q+1$ likelihood values $\mathcal{L}_i$ with i in $[0, q]$. If the prover is genuine, then the maximum value of likelihood should be $\mathcal{L}_0$ (since obtained using the session keys derived from the shared master key K) and a gap should exist between $\mathcal{L}_0$ and $\{\mathcal{L}_i\}_{i\in[1,q]}$ (denoted $\{\mathcal{L}_{i\neq 0}\}$ in the sequel). If the greatest likelihood value is not $\mathcal{L}_0$ or if $\mathcal{L}_0$ does not stand out from $\{\mathcal{L}_{i\neq 0}\}$, then the authentication is rejected and the protocol ends. Otherwise, the verifier continues the protocol by sending the ciphertext $C_k^{(0)}$ where k is chosen randomly in $[0, N-1]$.

Upon receiving $C_k^{(0)}$, the prover checks whether it belongs to the set of ciphertexts it has computed (*i.e.* whether it exists l in $[0, N-1]$ *s.t.* $C_l = C_k^{(0)}$) and if so, sends back a different ciphertext C_m (*i.e.* $m \neq l$). Otherwise, the prover rejects the authentication. The verifier finally checks whether it holds $C_l^{(0)}$ such that $C_l^{(0)} = C_m$. If it is the case the verifier and the prover are mutually and physically authenticated.

The most critical step for the success of our protocol is the acquisition of the physical leakage since it requires a perfect synchronization between the prover and the verifier. To ensure this, a trick would consist in triggering a timer, at the verifier side, upon sending the random value R_V. This timer considers the round-trip time between the prover and the verifier and the averaged time of executing the secure AES at the prover side. When the timer expires, the acquisition starts. Then, from the collected measurement the verifier needs to select the so-called Points Of Interest (POI) by applying some well-known selection algorithms (*e.g.* [**?**,**?**,**?**]). Following notations from Sec. 2.1, the verifier can target all the following bytes of AES states[7]:

$$\left.\begin{array}{ll} \text{AddRoundKey:} & X_r[l,c] \\ \text{SubBytes:} & Y_r[l,c] \\ \text{MixColumns:} & M_r[l,c] \end{array}\right\} \text{for } 0 \leq r \leq N_r - 1 \text{ and } (l,c) \in \{0, \cdots, 3\}^2 \quad (2)$$

[7] ShiftRows is not considered since it is simply a rearrangement of the AES state thus providing no additional information.

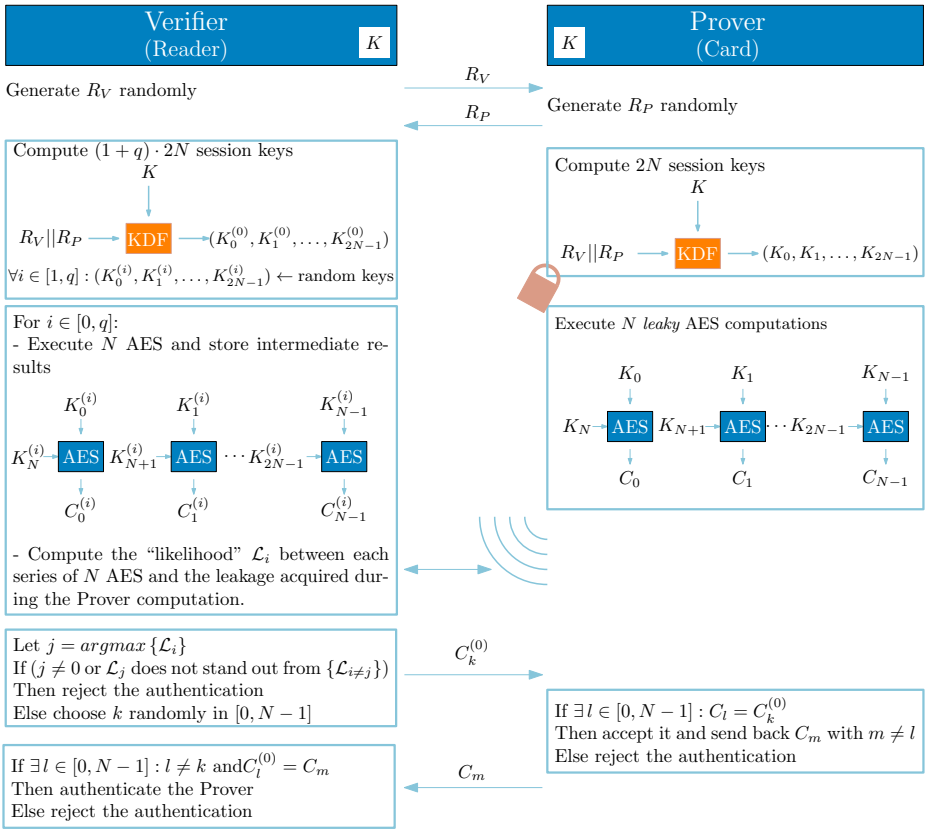


Fig. 1. Description of the physical authentication protocol.

Thus, each AES execution gives rise to $(3 \times 16 \times N_r)$ different sensitive byte variables along its execution. Each byte variable leaks following the same generic model (see Equation (1)) but instantiated with different model parameters (*i.e.* different pairs of (α_t, β_t)). Fortunately, it has been proven and confirmed experimentally that each sensitive byte variable leaks following the same model with the *same* parameters across the N AES executions.

Likelihood Computation. Following the notations from Algorithm 1, let $\{(X_r^{(j)}, Y_r^{(j)}, M_r^{(j)})_{0\leq r\leq N_r-1}\}$ be the set of internal states that the verifier predicts, round by round, when executing the j^{th} AES of the i^{th} set of N AES execution, *i.e.* $C_j^{(i)} = \text{AES}_{K_j^{(i)}}(K_{N+j}^{(i)})$. After computing the whole i^{th} set of N AES, the verifier holds $\{(X_r^{(j)}, Y_r^{(j)}, M_r^{(j)})_{(0\leq r\leq N_r-1)\times(0\leq j\leq N-1)}\}$. The computation of the i^{th} likelihood $\mathcal{L}_i$ can be executed following the paradigm illustrated in Algorithm 2.

Algorithm 2 Computation of the i^{th} likelihood $\mathcal{L}_i$

Require: $\{(X_r^{(j)}, Y_r^{(j)}, M_r^{(j)})_{(0 \leq r \leq N_r - 1) \times (0 \leq j \leq N-1)}\}$: the intermediate values, T: the acquired trace
Ensure: $\mathcal{L}_i$: the i^{th} likelihood value
1: $\mathcal{L}_i = 0$
2: **for each** $t = (l, c, r)$ **do**
3: Let $P_{X,t} = (X_r^{(j)}[l, c])_{0 \leq j \leq N-1}$
4: Let $L_{X,t} = \text{POI}_{X,t}(T)$
5: ▶ $\mathcal{L}_i$ += $\rho(P_{X,t}, L_{X,t})$ ▷ $\rho(.,.)$ denotes the Pearson correlation coefficient
6: Let $P_{Y,t} = (Y_r^{(j)}[l, c])_{0 \leq j \leq N-1}$
7: Let $L_{Y,t} = \text{POI}_{Y,t}(T)$
8: ▶ $\mathcal{L}_i$ += $\rho(P_{Y,t}, L_{Y,t})$
9: Let $P_{M,t} = (M_r^{(j)}[l, c])_{0 \leq j \leq N-1}$
10: Let $L_{M,t} = \text{POI}_{M,t}(T)$
11: ▶ $\mathcal{L}_i$ += $\rho(P_{M,t}, L_{M,t})$
12: **end for**
13: **return** $\frac{\mathcal{L}_i}{48 \cdot N_r}$ ▷ Normalize the outputted likelihood value

In Algorithm 2, for each sensitive byte variable uniquely determined by the triple $t = (l, c, r)$, the verifier constructs three vectors of predictions, namely $P_{X,t}$, $P_{Y,t}$ and $P_{M,t}$ and extracts three vectors of leakages, namely $L_{X,t}$, $L_{Y,t}$ and $L_{M,t}$ from the trace T acquired during the execution of the N leaky AES on the prover side. The extraction is done thanks to three POI selection functions $\text{POI}_{X,t}(T)$, $\text{POI}_{Y,t}(T)$ and $\text{POI}_{M,t}(T)$ which extract from the trace T the manipulation of resp. $(X_r^{(j)}[l, c])_{0 \leq j \leq N-1}$, $(Y_r^{(j)}[l, c])_{0 \leq j \leq N-1}$ and $(M_r^{(j)}[l, c])_{0 \leq j \leq N-1}$.

Test of Proximity. This section explains how the likelihood $\mathcal{L}_0$ compares to the set of likelihoods $\{\mathcal{L}_{i \neq 0}\}$ in order to detect malicious provers.

Let $\Lambda_{\neq 0}$ be the random variable which events are the likelihood values $\{\mathcal{L}_{i \neq 0}\}$, from which the verifier computes the estimated mean $\mu_{\neq 0}$ and mean deviation $w_{\neq 0} = (\max(\{\mathcal{L}_{i \neq 0}\}) - \mu_{\neq 0})$ of $\Lambda_{\neq 0}$. Consequently, to authenticate the prover, the verifier performs a test of proximity which consists in checking whether $\mathcal{L}_0 \geq \mu_{\neq 0} + 3 \cdot w_{\neq 0}$. This test allows the verifier to decide whether the likelihood value $\mathcal{L}_0$:

- is probably *not* an event from the random variable $\Lambda_{\neq 0}$, meaning that the prover probably owns the correct secret master key K or,
- is probably an event from the random variable $\Lambda_{\neq 0}$, revealing that the prover is a malicious one and thus leading the verifier to reject the physical authentication.

The choice of this test of proximity is inherited from the well-known 99% *level of confidence* of an event drawn from a Gaussian distribution. To obtain sufficiently good statistics, the number q of likelihoods computed using random keys should be at least 100.

The efficiency of this test is acknowledged by the experiments reported in Sec. 4.

3.3 Self-Assessment Evidence

From side-channel analysis perspective. The security of our protocol relies on the countermeasures that should be implemented to protect the secure KDF, which is can be based on the AES primitive for instance. To achieve this, one can take advantage of the several provably secure higher-order masking schemes that have been proposed in the literature [**?**,**?**,**?**]. So, the idea is to counteract an attacker trying to recover the shared master key by side-channel analysis.

From relay attack perspective. An attacker performing a relay attack must deal with the so-called round time trip which is the sum of two components of time: the processing time of the devices involved in the relay attack (genuine and malicious ones) and the communication times between all the involved devices. In the context of contactless cards, it is quite challenging to keep a low latency communication between contactless devices. Because of this non-negligible latency, the authors in [**?**] show that an attacker can connect a proxy-token to a mole with a high-speed link so that the communication time, denoted ϵ is close to 0 and thus defeating all classical distance-bounding protocols, i.e. protocols based on round time trip computations. The reason is that the processing times of malicious devices can be hidden in the error-time margins of the round time trip. Hopefully, still in [**?**], the authors show that using side-channel analysis allows to dramatically reduce this latency since the verifier can watch the prover in real time. So, the verifier is somehow plugged to the prover's brain while this latter is 'thinking', *i.e.* computing. In the protocol we proposed in Fig. 1, the verifier sends data blocks to the prover though the contactless interface. Once he has received the last expected acknowledgment from the prover, it can immediately plug to the prover's brain, *i.e.* start to acquire the prover's leakage induced by the cryptographic computations on the prover side. If the prover is a genuine one, the protocol will work properly while if it is a proxy-token one, the situation is clearly different. Indeed, denote by Δ the communication time required to exchange data between the verifier and a prover (genuine or not). If the prover is a fake one (*i.e.* a proxy-token), it is unavoidable for the attacker to spend a second communication time of Δ to exchange data between the mole and the genuine prover. Thus, even if the communication time ϵ between the proxy-token and the mole is negligible, the attacker relays the leakage from the genuine prover to the genuine verifier in time $2 \times \Delta$ instead of Δ when the genuine prover is in the proximity of the verifier. By doing so, the attacker relays a leakage which is desynchronized from the point of view of the genuine verifier. Therefore, the correlation coefficient computed on the verifier side will be too low to allow the authentication to succeed. Thus, as stated in Assumption 2, we considered as impractical for an attacker to relay physical leakage.

4 Experimental Validation

Experimental Setup. In our experiments, we considered a verifier that executes an AES-128 (*i.e.* $N_r = 10$) and targets only two internal states per AES round, namely the states after applying the AddRoundKey and SubBytes (these states are denoted by X and Y in Algorithm 1). Moreover, two different scenarios are studied. In the first one, the verifier focuses on the first round of each AES execution yielding 32 sensitive variables. In the second scenario, the verifier considers three rounds per AES execution yielding $32 \times 3 = 96$ targeted variables. The corresponding physical leakage was acquired using the ChipWhisperer-Lite board [**?**]. Furthermore, for robust statistics we instantiated $q = 1000$. The choice of the ChipWhisperer acquisition board is motivated by its simplicity and that the corresponding leakage model fits well our assumption in (1). Indeed, our protocol is intended for contactless device which implies that the corresponding leakage should be more noisy compared to the ChipWhisperer one. In the sequel, we consider this scenario by performing our experiments in a very noisy environment.

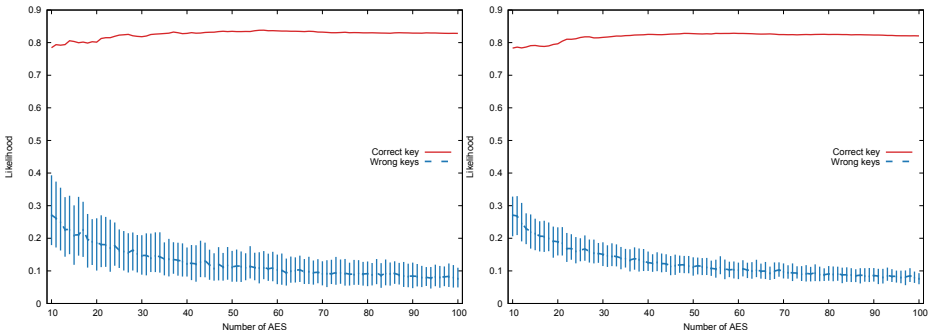
Assessment of our Proposal in a (almost) Noise-Free Environment. In the following, we considered a genuine prover (*i.e.* the acquired leakage corresponds to N AES executions using the correct session keys derived from the master key K). Then, for each scenario, we plotted the correct likelihood value $\mathcal{L}_0$ (the red curve) and the estimated mean $\mu_{\neq 0}$ and mean deviation $w_{\neq 0}$ of the random variable $\Lambda_{\neq 0}$ (the blue dotted curve and bars) according to an increasing number N of AES executions. The obtained results in both scenarios are illustrated in Fig. 2.

From Fig. 2, it is obvious that increasing the number N of executed AES enlarges the gap between $\mathcal{L}_0$ and $\{\mathcal{L}_{i\neq 0}\}$ when the prover is genuine. Moreover, one can conclude that the verifier is able to identify a genuine prover within merely $N = 10$ AES executions. Furthermore, considering more sensitive variables as shown in Fig. 2b, the gap between $\mathcal{L}_0$ and $\{\mathcal{L}_{i\neq 0}\}$ increases while the deviation margin decreases.

Assessment of our Proposal in a Noisy Environment. It is well-known that the ChipWhisperer-Lite is mostly noise-free[8]. To demonstrate the efficiency of our proposal in a noisy environment, the same acquired trace was reused to artificially increase the noise level (*i.e.* by adding a white Gaussian noise). We plotted in Fig. 3 the evolution of the acceptance rate according to an increasing noise standard deviation and a fixed number of executed AES ($N = 20$).

As expected, the acceptance rate decreases when the noise standard deviation increases. Moreover, for a fixed noise standard deviation, the more internal states, the higher the acceptance rate. Finally, we repeated the same experiments

[8] According to our measurements, it follows a Gaussian distribution with a standard deviation of ≈ 0.004.



(a) One round (b) Three rounds

Fig. 2. Evolution of the likelihood according to an increasing number of AES executions.

when increasing the number of executed AES and used the Signal-to-Noise Ratio (SNR) rather than the noise standard deviation to better quantify the amount of white noise which is added. The obtained results are depicted in Fig. 4.

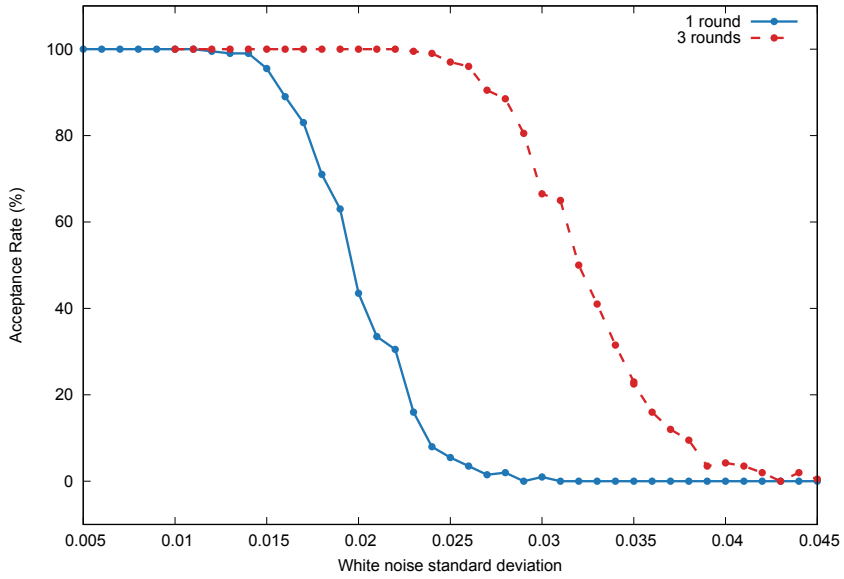
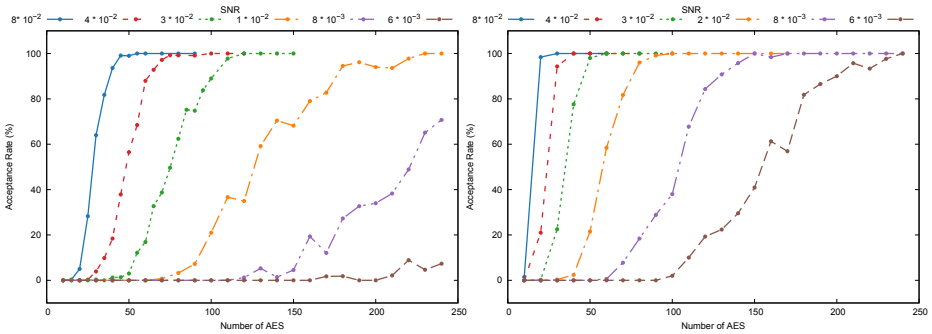


Fig. 3. Evolution of the acceptance rate according to an increasing noise standard deviation and a fixed number of executed AES.

From Fig. 4, one can conclude that in the presence of a significant amount of noise, increasing the number N of AES executions allows the verifier to better



(a) One round (b) Three rounds

Fig. 4. Evolution of the acceptance rate according to an increasing number of executed AES and SNR.

distinguish the presence of a genuine prover. Moreover, the number of variables used for the likelihood computation is also of great interest. Indeed, by considering more AES rounds and more internal states per round, the verifier increases the acceptance rate while executing less AES computations.

Regarding the choice of the optimal parameters (*i.e.* number of AES, number of rounds, ...), it is up to the designer to choose the suitable ones with respect to the the noise level and the required acceptance rate.

FAR and FRR Assessment. In this section, we evaluated our proposed authentication protocol with respect to two well-known security metrics: the *False Acceptance Rate* (FAR) and the *False Rejection Rate* (FRR). The FAR is the measure of probability that an authentication protocol accepts an unauthorized prover while the FRR is the measure of probability that an authentication protocol rejects a genuine prover.

To do so, we considered in the following an authentication between a verifier and a malicious prover. We acquired one trace on the ChipWhisperer board corresponding to N executions of leaky AES using wrong session keys at the prover side. We stress the fact that the noise level is similar to the experiments reported in Fig.2b (*i.e.* $\sigma \approx 0.004$). Then, we plotted in Fig. 5 the evolution of the likelihood $\mathcal{L}_0$ (the red curve) and the estimated mean and mean deviation $(\mu_{\neq 0}, w_{\neq 0})$ of $\Lambda_{\neq 0}$ (the blue dotted curve and bars) according to an increasing number of executed AES when considering three rounds per AES execution.

From Fig. 5, one can see that the curves overlap and hence the verifier concludes that the prover is a malicious one and rejects the authentication. So, the obtained results confirm that the FAR of our proposed authentication protocol is almost zero independently of the number of executed AES. We recall that, by design, the theoretical FAR is 2^{-128} which the probability that a malicious prover guesses correctly the 128-bit shared master key K.

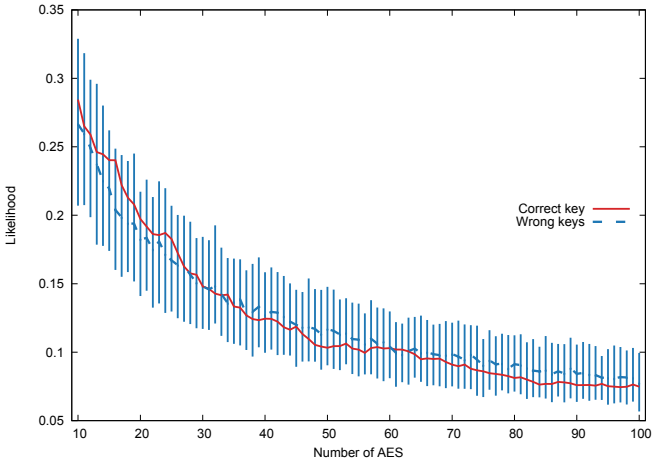


Fig. 5. Evolution of the likelihood according to an increasing number of AES executions when considering a malicious prover.

Regarding the FRR, we demonstrated in Fig. 4 that in the presence of a genuine prover the acceptance rate decreases (*i.e.* the FRR increases) when the noise standard deviation increases. On the other hand, when the number of executed AES increases and/or when considering several intermediate values then the acceptance rate increases too. This implies that, depending on the application, one can adjust the FRR by tuning the number of executed AES and the number of targeted intermediate values with respect to the noise level. This can be done as a beforehand agreement between the genuine prover and the verifier.

5 Conclusion

This work highlights the bright side of side-channel leakage. The traditional doctrine has always exhibited this leakage as a serious practical threat to cryptographic embedded systems. In this paper, we have introduced a new authentication protocol that constructively exploits the side-channel leakage to prevent relay attacks. To be authenticated, the prover performs some cryptographic operations using a beforehand shared master key (either loaded during the personalization phase of the prover or more simply by using the classical Diffie-Hellman key exchange protocol). The verifier eavesdrops the resulting physical leakage and computes the likelihood using the corresponding theoretical predictions. When the resulting likelihood is quite high (with respect to the considered proximity criterion) then the prover could be accepted after a last step of validation by exchanging some ciphertexts. We have argued and confirmed with experiments, that our proposal is secure against both side-channel attacks and relay attacks. Moreover, it solves some security and design issues pinpointed in a previous

study [**?**]. Besides, through our experimental validation, we have demonstrated the efficiency of our proposal in terms of FAR and FRR.

In view of these promising results, a natural open problem is to validate our proposal in an even more realistic scenario by exploiting for example the electromagnetic leakage captured on a modern smart-card chip.