Effective and Efficient Masking with Low Noise using Small-Mersenne-Prime Ciphers

Loïc Masure[1][0000−0003−2978−4067], Pierrick Méaux[2][0000−0001−5733−4341], Thorben Moos[1][0000−0003−3809−9803], François-Xavier Standaert[1][0000−0001−7444−0285]

[1] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium.
[2] Luxembourg University, SnT, Luxembourg.

Abstract. Embedded devices used in security applications are natural targets for physical attacks. Thus, enhancing their side-channel resistance is an important research challenge. A standard solution for this purpose is the use of Boolean masking schemes, as they are well adapted to current block ciphers with efficient bitslice representations. Boolean masking guarantees that the security of an implementation grows exponentially in the number of shares under the assumption that leakages are sufficiently noisy (and independent). Unfortunately, it has been shown that this noise assumption is hardly met on low-end devices. In this paper, we therefore investigate techniques to mask cryptographic algorithms in such a way that their resistance can survive an almost complete lack of noise. Building on seed theoretical results of Dziembowski et al., we put forward that arithmetic encodings in prime fields can reach this goal. We first exhibit the gains that such encodings lead to thanks to a simulated information theoretic analysis of their leakage (with up to six shares). We then provide figures showing that on platforms where optimized arithmetic adders and multipliers are readily available (i.e., most MCUs and FPGAs), performing masked operations in small to medium Mersenne-prime fields as opposed to binary extension fields will not lead to notable implementation overheads. We compile these observations into a new `AES`-like block cipher, called `AES-prime`, which is well-suited to illustrate the remarkable advantages of masking in prime fields. We also confirm the practical relevance of our findings by evaluating concrete software (ARM Cortex-M3) and hardware (Xilinx Spartan-6) implementations. Our experimental results show that security gains over Boolean masking (and, more generally, binary encodings) can reach orders of magnitude despite the same amount of information being leaked per share.

1 Introduction

Research question. Masking is an important countermeasure against side-channel attacks. Introduced in [46,27], it has attracted significant attention thanks to the strong security guarantees it can provide [53,76,36,37]. Since leading to efficient implementations in software [79,13], bitslice software [47,49] and hardware [50,23], additive (Boolean) masking is for now the most investigated type of encoding. Concretely, assuming that the shares' leakage is sufficiently

noisy and independent, Boolean masking can amplify the noise of an implementation (and therefore its security) exponentially in the number of shares.

Yet, and despite these strong theoretical guarantees, ensuring the noise and independence conditions may not be easy in practice. The independence issue is a well investigated one. Physical defaults such as glitches [61,62] or transitions [30,8] can cause leakage about re-combined shares. Fortunately, these defaults can be circumvented (at some cost) thanks to well understood design techniques [72,44,24]. To the best of our knowledge, the noise issue is for now a less investigated one. Concrete results of so-called horizontal attacks such as [12,20] showed that a lack of noise can lead to devastating attacks against Boolean masking. Improved security against horizontal attacks has been captured with the notion of noise rate [5,25]. But gadgets with limited noise rate only reduce the number of manipulations of the shares (in order to prevent reducing the noise by averaging). Therefore, they have limited impact when the noise level of an implementation is already small without averaging, as it is for example the case for small embedded devices (e.g., 32-bit ARM Cortex or similar cores). Another class of attacks which has been shown to threaten the security of Boolean masking by exploiting an insufficient noise level is based on leveraging the static power consumption of devices as a side-channel. In such attacks which, unlike horizontal attacks, are mostly a concern for hardware implementations, the adversary obtains the leakage of a halted computation state with almost arbitrarily low noise, which limits the effectiveness of Boolean encodings [70,75,68,66,67].

As a result, the main objective of this paper is to initiate a study of encodings and ciphers that can lead to secure and efficient low-noise implementations. Precisely, we question the possibility that increasing the number of shares in a masking scheme leads to security amplification without any noise.

Seed results. Interestingly, the literature contains several hints that the answer to this question might be positive if changing the Boolean encoding into a more "complex" one. On the one hand, it has been observed that the Inner Product (IP) masking introduced in [40] can lead to significantly better security in low-noise settings than Boolean masking, for example in case of leakage functions that are close to the Hamming weight one (see for example [7], Figure 3 for an illustration with two shares). Unfortunately, computing on IP encodings generally leads to significant implementation overheads, and it remains an open question whether its security vs. efficiency trade-off can compare positively with the (simpler) multiplication algorithms of additive masking schemes.

On the other hand, Dziembowski et al. showed that the level of noise required for masking to amplify security can be significantly reduced if the encodings are defined in groups of prime orders [41]. Such an observation has the significant advantage of being valid for simple (additive) masking schemes that have been intensively studied in recent years and can benefit from increasingly efficient automated verification tools [10,17,9,57]. However, the practical impact of these seed investigations has not been studied yet, leaving open questions like:

– How do these theoretical guarantees translate into concrete security guarantees for practically-relevant leakage functions and noise levels?

– What is the impact of increasing the prime size in these practical cases?
– Can masking with prime encodings be used to improve the side-channel security of block cipher implementations in software and hardware?
– What is the (software and hardware) cost of implementing masked block cipher operations in prime fields instead of binary extension fields?

Contributions. Based on this state of the art, we pick up on the challenge of better understanding masking in low-noise settings and propose encodings and ciphers that allow secure and efficient implementations in this context. More precisely, our contributions in this respect are in three parts:

First, we show in Section 2 that moving from Boolean encodings to arithmetic encodings, first in binary fields, then in prime fields, leads to gradual side-channel security improvements. We use the information theoretic framework put forward in [84] for this purpose, and consider both the standard Hamming weight leakage model and a (localized) model leaking the Least Significant Bit (LSB) of the target intermediate computations in our evaluations. It allows us to confirm Dziembowski et al.'s claimed gains in low-noise environments, but also to observe that these gains can be maintained without any noise (for these non-injective leakage functions) and are preserved as the noise increases. We additionally explain the low-noise weakness of Boolean encodings formally and show that for the practically-relevant Hamming weight leakage function, increasing the size of the prime moduli improves the side-channel security of masked encodings.

Next, we consider the question of efficiency. In Section 3, we show that by selecting small Mersenne primes to operate our masked computations, it is possible to implement them with performances that compare with binary fields, especially in case optimized arithmetic adders and multipliers are available (e.g., on most recent MCUs and FPGAs). Since standard symmetric designs are not directly suitable for efficient masked implementations with non-binary encodings, we then consider so-called `prime` ciphers in Section 4. A `prime` cipher is a cipher which performs all operations in $\mathbb{F}_p$ with p a prime modulus. In order to illustrate our results, we then consider `AES-prime` as a first example of such a `prime` cipher, where the S-box is based on a small power in $\mathbb{F}_p$ and the MixColumns operation is based on an MDS matrix in the same field.

Eventually, we move to the concrete evaluation of our designs in Section 5. Since our simulated evaluations in Section 2 are based on the Hamming weight and LSB leakage functions, and are limited to encodings, an important question is whether the security guarantees of these examples are observed when measuring concretely-relevant implementations with several exploitable target sensitive operations that may not exactly leak as assumed in our simulations. We answer this question positively by experimentally analyzing software (ARM Cortex-M3) implementations, where the (worst-case) adversary is first given full profiling access to the device to characterize its leakage behavior before performing the actual attack. Our results confirm that masking `prime` ciphers with prime encodings can significantly improve the security compared to Boolean masked

designs in a low-noise setting. We also conduct a hardware (FPGA) case study, confirming the improved security provided for a naturally noisier target.

Cautionary note: why the `AES-prime`? Initiating the investigation of new encodings naturally raises the question of what is the best cipher for evaluating them. As for example witnessed by the NIST Lightweight cryptography competition, the vast majority of the state-of-the-art ciphers designed for masking are bitslice ones.[1] Unfortunately, such ciphers cannot be easily turned into "prime equivalents". At the opposite side of the spectrum, the use of large prime moduli has recently attracted a lot of attention for the design of ciphers tailored for advanced cryptographic applications (e.g., multiparty computation, hybrid fully homomorphic encryption or zero knowledge proofs). Examples include `MiMC` [2] and its Feistel variant `GMiMC` [1], `Rescue` [4], `HADESMiMC` [48], `CIMINION` [34], `HERA` [28] or `PASTA` [35]. In general, these ciphers are not directly adapted to our goals either, since their proposed instances usually favor multiplications in large fields (in order to reduce their overall number) while embedded implementations crucially require small and well-chosen primes for efficiency. As a result, there is also little work on the secure implementation of such new ciphers. Given this state of the art, we turn back to `AES`-like ciphers for which it is easy to specify binary and prime versions. This allows us to leverage the wide body of research on countermeasures and evaluation tools tailored for the `AES` (which, we hope, can further stimulate external analyzes and follow up studies). It also allows us to work with primes that are well suited to the software and hardware implementations we target. We insist that the goal of the `AES-prime` cipher is only to illustrate the potential of prime masking. Since illustrating this potential requires mixing abstractions from different research fields, we admittedly do not claim that its security analysis is as comprehensive as if the very design of the `AES-prime` was our main contribution. So the security analysis we provide is only aimed to show that a prime cipher with an `AES`-like structure can be secure with a similar number of rounds as a binary cipher with an `AES`-like structure, based on the (standard) cryptographic properties of its components. Overall, the `AES-prime` may not be the best cipher for prime masking in the long run, but it is a suitable starting point for a comparison, since `AES` and `AES-prime` are the closest match between binary and `prime` cipher that we have at the moment. We hope that the promising results it leads to can motivate the design of new ciphers that are tailored for this specific application of prime-field masking and can compete with bitslice ciphers from an efficiency point of view.

2 From Boolean to prime field arithmetic masking

In this first section, we revisit the theoretical investigations of Dziembowsi et al. from a more practical viewpoint. Precisely, it is shown in [41] that masking with encodings in prime fields can lead to effective noise amplification. We next question the concrete security that can be observed for practically-relevant leakage

[1] https://csrc.nist.gov/Projects/lightweight-cryptography.

functions without any noise, and whether the gains of these prime encodings are maintained in high noise regimes. We additionally show the positive impact of increasing the size of the prime moduli and provide theoretical insights on our results and their generalization to parallel implementations.

2.1 Methodology

As a usual starting point to analyze the worst-case security provided by a countermeasure quantitatively, we use the information theoretic framework put forward in [84]. Namely, we will compute the mutual information between a target sensitive value $X \in \mathcal{X}$ and the leakage of its shares $\boldsymbol{L}$, that is:

$$\mathrm{MI}(X; \boldsymbol{L}) = \mathrm{H}(X) + \sum_{x \in \mathcal{X}} \mathsf{p}(x) \cdot \int_{\boldsymbol{l} \in \mathcal{L}^d} \mathsf{f}(\boldsymbol{l}|x) \cdot \log_2 \mathsf{p}(x|\boldsymbol{l}), \tag{1}$$

with $\mathsf{p}(x)$ the shortcut notation for $\Pr(X = x)$. Assuming uniformly distributed sensitive values, $\mathrm{H}(X) = \log_2(|\mathcal{X}|)$ and $\mathsf{p}(x|\boldsymbol{l})$ is computed as $\frac{\mathsf{f}(\boldsymbol{l}|x)}{\sum_{x^* \in \mathcal{X}} \mathsf{f}(\boldsymbol{l}|x^*)}$ where $\mathsf{f}(\boldsymbol{l}|x)$ is the Probability Density Function (PDF) of the leakage samples. In the case of a masked implementation with d shares, this PDF then takes the shape of a mixture distribution defined as $\mathsf{f}(\boldsymbol{l}|x) = \sum_{r \in \mathcal{X}^{d-1}} \mathsf{f}(\boldsymbol{l}|x, r) \cdot \mathsf{p}(r)$.

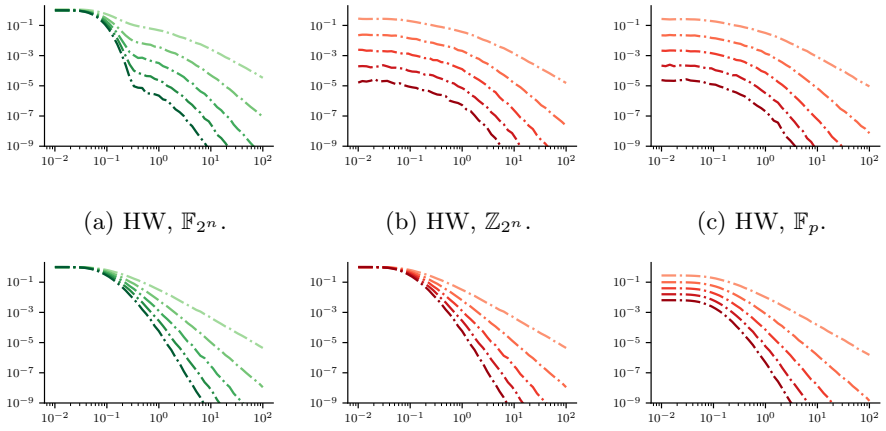
In the following, we will make the standard assumption that the leakage of each component $\mathsf{p}(\boldsymbol{l}|x, r)$ in the mixtures is a Gaussian distribution, so that the leakage of each share can be written as $\mathsf{L}(X_i) = \delta(X_i) + N_i$, the full leakage vector can be written as $\boldsymbol{L} = (\mathsf{L}(X_1), \mathsf{L}(X_2), \ldots, \mathsf{L}(X_d))$ and the variance of the noise σ^2 is a security parameter. As for the deterministic part of the leakage function δ, we will consider both the standard Hamming weight function and a (more localized) bit leakage function leaking the LSB of X_i.

Note that directly computing the mutual information rapidly turns out to be computationally intensive as the number of shares increases. This is for example witnessed by the results of Fumaroli et al. [45, Fig. 2] and Standaert et al. [85, Fig. 7] which were limited to $d \leq 3$. We improve over these previous works by leveraging the fact that computing the mixture PDF of a masked encoding can be done without summing over all the terms of the mixture explicitly, because the leakage of such masked encodings can be written as a convolution product [64, Prop. 1]. Moreover, if several encodings of sensitive intermediate computations leak, the latter observation can be generalized as a Soft Analytical Side-Channel Attack (SASCA) without cycles [51,20], where the Belief Propagation (BP) algorithm efficiently provides an exact solution [86]. Therefore, the complexity of evaluating Equation 1 actually scales in $\mathcal{O}(d \cdot n \cdot 2^n)$, instead of $\mathcal{O}\left(2^{2n \cdot d}\right)$ for a naive approach. Concretely, we use the SCALib library for this purpose.[2] It allows us to analyze the leakage of up to 13-bit targets with up to 6 shares.

2.2 Information theoretic evaluation results

The results of our information theoretic investigations are depicted in Figure 1. Recall that the number of traces to perform a key recovery attack is inversely

[2] https://scalib.readthedocs.io/en/latest/index.html



(a) HW, $\mathbb{F}_{2^n}$. (b) HW, $\mathbb{Z}_{2^n}$. (c) HW, $\mathbb{F}_p$.

(d) LSB, $\mathbb{F}_{2^n}$. (e) LSB, $\mathbb{Z}_{2^n}$. (f) LSB, $\mathbb{F}_p$.

Fig. 1: Information theoretic evaluation of different masked encodings for different numbers of shares. Top: Hamming weight leakage function, bottom: LSB leakage function. Left: Boolean encoding, middle: arithmetic encoding in $\mathbb{Z}_{2^n}$, right: arithmetic masking in $\mathbb{F}_p$. The X axis is the noise variance in log scale. The Y axis is the MI in log scale. The different curves are for increasing numbers of shares (from 2 to 6). The target sensitive variable is on 8 bits.

proportional to the MI [33]. So as expected in theory, all these curves have a slope $-d$ in the high noise regime [63]. The relevant observations for our investigations are twofold. First, arithmetic masking significantly improves the situation in low-noise regimes. This is reflected by the "stepped" regions of the curves. For Boolean masking, the left (low noise) parts of the figure show no reduction of the MI when increasing the number of shares. By contrast, arithmetic masking can lead to (exponential) security improvements (i.e., equidistant steps) in the same region. This only holds for the Hamming weight leakage function (on the top of the figure) when considering arithmetic masking in $\mathbb{Z}_{2^n}$ (in the middle plots) and it even holds for the LSB leakage function (on the bottom of the figure) when considering arithmetic masking in $\mathbb{F}_p$ (in the right plots). We insist that this exponential security amplification without noise of prime masking theoretically holds for any non-injective leakage function [41]. Our evaluations amplify this fact with the reassuring observation that its concrete impact is especially strong with leakage functions that are commonly considered to be suitable abstractions of real device behavior. Second, the gains that are obtained with low noise are maintained when increasing the noise, which was not studied by Dziembowski et al. So these results confirm that there is an interest to use prime encodings for better dealing with noise-free leakages, and put forward that such encodings can also lead to significant security improvements when leakages become noisy.

2.3 Theoretical explanation

We now argue why increasing the number of Boolean shares without noise is useless in presence of Hamming weight leakage. For this purpose, we use a spectral analysis of the conditional Probability Mass Functions (PMFs) spanned by the noise-free Hamming weight leakage model. We said in subsection 2.2 that such distributions can be computed through discrete convolutions [64, Prop. 1]. So according to the convolution theorem, each PMF $\mathsf{p}(X|\boldsymbol{L})$ can be computed in a transformed domain as the element-wise product of the d PMFs — expressed themselves in the same transformed domain — associated to each of the corresponding shares $\mathsf{p}(X_i|L_i)$. For Boolean masking, this transformed domain is described by the *Walsh-Hadamard* transform over the input domain $\mathbb{F}_2^n$, which can be seen as an n-dimensional Fourier transform over $\mathbb{F}_2$. Therefore, the ω-th coefficient of the Walsh-Hadamard transform is computed as:

$$\mathsf{WHT}(\mathsf{p},\omega) = \sum_{x\in\mathbb{F}_2^n} (-1)^{\langle\omega,x\rangle}\mathsf{p}(x|l), \tag{2}$$

where $\langle\omega,x\rangle$ is the inner product between ω and x. Figure 2 below depicts it when computed for Hamming weight leakages corresponding to a 4-bit target variable. It can be observed that for $l=0$ or $l=n$ (i.e., the dotted gray curves

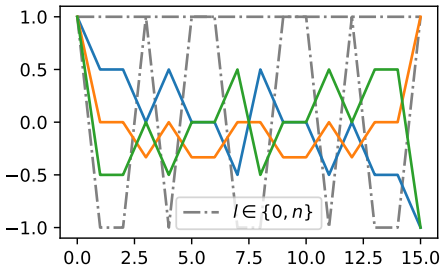


Fig. 2: Walsh-Hadamard transform of the conditional distribution $\mathsf{p}(X|\boldsymbol{L})$ for each hypothetical value of the Hamming weight leakage function.

in Figure 2), the absolute value of the Walsh-Hadamard coefficients is a constant 1. This corresponds to values for which the leakage model is injective (i.e., the conditional probability distribution of the sensitive variable collapses to a single Dirac). Let us first consider the unrealistic assumption that these leakages are never observed – we will discuss the general case afterwards. For the remaining leakages that an adversary may observe, only the first and last coefficient of the Walsh-Hadamard transform are equal to 1 in absolute value. The first coefficient being equal to 1 is due to p being a probability distribution (i.e., all probabilities are summing to 1). For the last coefficient (i.e., for $\omega = 1^n$), the inner product $\langle 1^n, x\rangle$ coincides with the Hamming weight of x, *i.e.* $\mathrm{HW}(x) = \langle 1^n, x\rangle$. Hence:

$$\mathsf{WHT}(\mathsf{p},1^n) = \sum_{x\in\mathbb{F}_2^n} (-1)^{\langle 1^n,x\rangle}\mathsf{p}(x|l) = \sum_{x\in\mathbb{F}_2^n} (-1)^{\mathrm{HW}(x)}\frac{\mathbf{1}_{\mathrm{HW}(x)=l}}{\binom{n}{l}} = (-1)^l. \tag{3}$$

As a consequence, any element-wise product between d Walsh-Hadamard transforms among the ones observable in a Hamming weight leakage model can only decrease all coefficients at an exponential rate, except the first and last ones. It results that when $d \to \infty$, the Walsh-Hadamard transform of the masked leakage distribution tends towards $(1, 0, \ldots, 0, \pm 1)$. Such asymptotic Walsh-Hadamard transforms correspond to uniform distributions over two non-overlapping supports of equal size 2^{n-1}, both leading to a conditional entropy of $n-1$ bits.

Finally, we discuss our assumption that the adversary did not observe any sample such that $l = 0$ or $l = n$. As a consequence, d in our previous reasoning is replaced by the number of samples in the leakage that are neither null nor equal to n. Let us denote this number by the random variable T. Hereupon, we may notice that the marginal distribution of the leakage is such that T follows a binomial law of parameter $\mathcal{B}(d, \frac{1}{2^{n-1}})$. Such a law is known to concentrate exponentially fast towards its mean $\frac{d}{2^{n-1}}$. As a result, the probability to observe a number of null or full (equal to n) leakages becomes negligible when $d \to \infty$.

2.4 Intuitive explanation

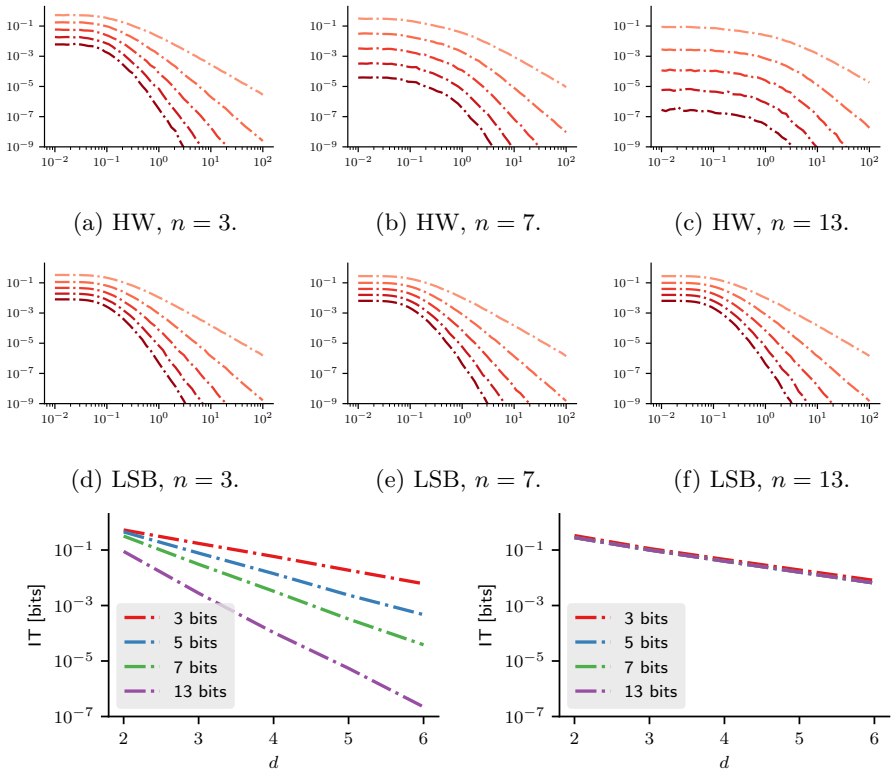
The theoretical explanation confirms our observations from the information theoretic analysis formally. To gain a more intuitive understanding of why the security level stagnates for binary masking (Boolean and arithmetic) when increasing the number of shares without noise we can also point to concrete properties of the considered leakage functions. If an adversary receives noise-free Hamming weight observations HW(x_1), ..., HW(x_n) of the Boolean shares x_1, ..., x_n of a secret variable x with $x_1 \oplus ... \oplus x_n = x$, then the parity-bit $b \equiv$ HW(x_1) + ... + HW(x_n) mod 2 is also the parity of the Hamming weight of x. Likewise, the parity of noise-free LSB observations of all shares is also the LSB of the secret, since LSB(x) $\equiv$ LSB(x_1) + ... + LSB(x_n) mod 2. The latter equation holds for both Boolean masking and arithmetic encodings in binary fields. In all described cases the information learned about the secret variable is independent of the number of shares and the statistical security order. The order of the masking only becomes relevant when increasing the noise level. By contrast, for arithmetic encodings in prime fields no such relationships exist and an exponential decrease of the MI can be observed even in the no-noise scenario. This is true for any non-injective leakage model (i.e., in any case where not all shares and intermediates are already known to the adversary with probability 1).

2.5 Impact of the prime size

The results in Section 2.2 are for 8-bit targets. A natural further question is whether increasing the size of the prime modulus has any (positive or detrimental) effect on security. This investigation is especially interesting since the field size is a source of potential non-tightness in masking security proofs [54,63].

Using the same information theoretic approach as in Section 2.2, we can observe in Figure 3 (especially in the bottom parts of the figure) that increasing the prime size significantly improves the security of the masked encodings for the

Hamming weight leakage function, while it has no impact for the LSB leakage function.[3] Again, this is a quite positive outcome since the Hamming weight leakage function is commonly considered as a reasonable simplification of many leakage functions observed in practice. It also recalls that side-channel security against very localized leakage functions (e.g., the LSB one that corresponds to a probing attack) is very challenging to obtain. But as observed in [58], such models generally exploit significantly more powerful (and expensive) sources of leakage than the power consumption or electromagnetic radiation.



(a) HW, $n = 3$. (b) HW, $n = 7$. (c) HW, $n = 13$.

(d) LSB, $n = 3$. (e) LSB, $n = 7$. (f) LSB, $n = 13$.

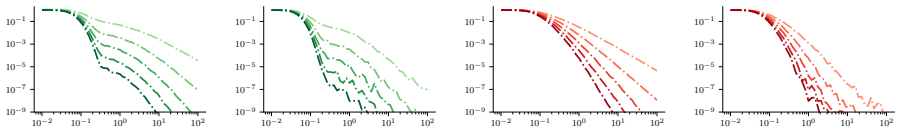
(g) Noiseless HW leakage function (h) Noiseless LSB leakage function

Fig. 3: Information theoretic evaluation of masked encodings with prime moduli $p = 2^n - 1$ of increasing sizes. Top: Hamming weight leakage function. Middle: LSB leakage function. The bottom figures are for the no-noise regime only.

[3] In fact, for the LSB leakage function and a fixed number of shares, it can even be shown that the $\mathrm{MI}(X; \boldsymbol{L})$ is lower bounded when increasing the size of p. For 2 shares the concrete lower bound is given by $\lim_{p \to \infty} \mathrm{MI}(X; \boldsymbol{L}) = 0.2787$.

2.6 Parallel leakage

By assuming that the adversary can observe the leakage of each share separately, our previous evaluations naturally correspond to a serial (e.g., software) implementation. Yet, the problem that we observe with low physical noise leakages actually holds even in the case of a parallel manipulation of the shares (more reflective of a hardware implementation). We analyzed this scenario by simply replacing the leakage vector $\boldsymbol{L}$ by the sum of its d elements. The resulting evaluation is depicted in Figure 4 (for 8-bit targets). As expected, we see that the



(a) HW, serial, $\mathbb{F}_{2^n}$. (b) HW, par., $\mathbb{F}_{2^n}$. (c) LSB, serial, $\mathbb{Z}_{2^n}$. (d) LSB, par., $\mathbb{Z}_{2^n}$.

Fig. 4: Information theoretic evaluation of serial and parallel binary masking.

curves of the parallel implementation are always below (i.e., less informative) than the ones of the serial implementation, which is explained by the accumulation of the leakage of the shares processed in parallel, which leads to a loss of information available to the observer. More interestingly for our following investigations, we also see that the curves in Figure 4b and Figure 4d are stuck to the 1 bit threshold in the low-noise regime, matching the observations in [82].

Here as well, our previous spectral analysis provides an explanation of our observations. In the serial case, the adversary is given a leakage tuple $\boldsymbol{l} = (l_1, \ldots, l_d)$, so the PMF $x \mapsto \mathsf{p}(X = x|\boldsymbol{l})$ is the convolution product $x \mapsto (\mathsf{p}(X_1|l_1) * \ldots * \mathsf{p}(X_d|l_d))(x)$, as previously discussed [64, Prop.1]. In the parallel case, the adversary is only given the sum $\ell = \sum_i l_i$ of the tuple $\boldsymbol{l}$, also denoted by $\mathcal{S}(\boldsymbol{l})$ hereafter. By applying the total probability formula over all the tuples $\boldsymbol{l}$ verifying $\mathcal{S}(\boldsymbol{l}) = \ell$, and leveraging the mutual independence of the variables $L_1, \ldots, L_d$, the conditional PMF $\Pr(X|\mathcal{S}(\boldsymbol{L}))$ verifies:

$$\Pr(X|\mathcal{S}(\boldsymbol{L}) = \ell) = \sum_{\boldsymbol{l}:\mathcal{S}(\boldsymbol{l})=\ell} (\mathsf{p}(X_1|l_1) * \ldots * \mathsf{p}(X_d|l_d)) \cdot \Pr(\boldsymbol{L} = \boldsymbol{l}|\mathcal{S}(\boldsymbol{l}) = \ell) \quad . \quad (4)$$

In other words, the PMF becomes the averaged convolution product over all possible tuples verifying the constraint. Nevertheless, we argue that when $d \to \infty$, averaging does not affect the resulting PMF much. Indeed, for each tuple verifying the constraint, the parity of the number of odd values remains constant: if ℓ is even, there is always an even number of odd values in the tuple $\boldsymbol{l}$. Likewise, if ℓ is odd, there is always an odd number of odd values in the tuple $\boldsymbol{l}$. This ensures that all the convolution products that are averaged converge towards the same uniform distribution over a subset of size 2^{n-1}, as argued in subsection 2.3. Hence, the resulting conditional entropy (of $n-1$ bits) remains unaffected.

2.7 Final remark

While arithmetic masking in prime fields can be *sufficient* to deal with low-noise leakages, we may wonder whether it is *necessary* to consider groups of prime size, or if an odd modulus would suffice. This case has been discussed by Dziembowski et al. who showed that for groups of composite order, there exists some leakage models for which masking is ineffective [41, Prop. 1]. This result is actually closely linked to a wide literature studying the convergence of probability distributions through the iterative application of a self-convolution product [56,3]. For example, consider the group $\mathbb{Z}_{15}$, where the inner law is the addition modulo 15. Assuming that the target variable leaks such that $\mathsf{L}(x) = x \bmod 3$, the adversary will obtain $\log_2(3)$ bits. It can be verified that masking at any order keeps the conditional probability distribution of the target variable unchanged, up to a permutation, which in turn keeps the mutual information constant. Using prime orders avoids this theoretical possibility. As will be clear in Section 4, it also makes the cryptanalytic treatment of `prime` ciphers easier.

3 Performance and cost

The primary motivation to tailor block ciphers and masking schemes towards binary fields is efficiency. From an implementation perspective the ability to perform field addition/subtraction using a simple bit-wise Exclusive-OR (XOR) operation is one of the core advantages of working in $\mathbb{F}_{2^n}$ compared to performing the equivalent tasks in $\mathbb{F}_p$. Field multiplication including the reduction using an irreducible polynomial can be implemented quite efficiently for small n as well. In hardware, XOR/AND sequences are used for this purpose, in software *log/alog* tables are one of the most efficient options [32]. In this section we argue that prime field arithmetic can be executed with similar (and sometimes even better) efficiency as (than) binary arithmetic on many platforms, due to a direct utilization of existing computation structures. In fact, devices like Micro-Controller Units (MCUs) and Field-Programmable Gate Arrays (FPGAs) are mostly developed with general purpose computing in mind. Hence, they often provide regular arithmetic operations like addition, subtraction and multiplication as dedicated and heavily-optimized hardware circuitry (for one specific size of operands). Many 32-bit micro-controllers for example include single-cycle arithmetic instructions for 32-bit operands. Yet, single-cycle multiplications sometimes produce only a 32-bit result, instead of a full 64-bit product (e.g., ARM Cortex-M0/M3). FPGAs on the other hand offer DSP slices which commonly include full 18×18-bit or 27×18-bit multipliers. Whenever such a hardware support is available, the heavy lifting for implementing prime field arithmetic is (at least for small primes) taken care of at the expense of occupying the integrated arithmetic accelerators (temporarily). The remaining element that might be a bottleneck in such implementations is the modular reduction. Yet, the efficiency of reduction algorithms modulo a prime is a well-studied subject in general and in Elliptic-Curve Cryptography (ECC) in particular.

3.1 Small Mersenne primes

It is commonly known that reduction modulo a Mersenne prime, i.e., a prime of the form $p = 2^n - 1$, can be performed very efficiently on a binary computer. There are further categories of primes that emerged as particularly suitable choices for efficient modular reduction. These include generalized Mersenne primes, pseudo-Mersenne primes and Montgomery-friendly primes [52,18,19,6]. Yet, these alternatives are mostly needed because large Mersenne primes are sparsely distributed. In the range between $2^{127} - 1$ and $2^{521} - 1$ for example, there exist none. For our purposes, however, we are primarily interested in primes much smaller than that, namely with bit-lengths close to the size of binary extension fields that popular symmetric block ciphers operate in, e.g., $\mathbb{F}_{2^8}$ for the AES [32]. The Mersenne prime exponents closest to 8 are $n = 7$ and $n = 13$. In the following we extend this range a bit and compare the performance and cost of masked multiplication algorithms in fields $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^n-1}$ for all Mersenne exponents n with $3 \leq n \leq 31$. These sizes allow efficient implementation of field arithmetic both in software and hardware. Hence, they are relevant targets for the construction of cryptographic building blocks. Besides, Mersenne prime fields have the additional advantage that any multiplication of a field element by a power of 2 is merely a rotation of the bits, which is cheap in software and entirely free in hardware. As a result, also the Hamming weight of a value is preserved when it is multiplied by a power of 2. Eventually, when mapping messages into the desired prime field for encryption, Mersenne primes cause the minimum amount of unused bit strings (i.e., only the all-ones string).

3.2 Masked multiplication in binary fields vs. prime fields

When masking a cryptographic primitive like a block cipher, the linear operations can trivially be extended and applied to each share individually. Therefore, the cost of masked linear operations grows linearly in the number of shares. The implementation of non-linear elements is less straightforward and requires dedicated gadgets which optimally offer (robust) probing security and, if desired, satisfy composability notions to enable their secure combination to construct larger (robust) probing secure circuits [44,11]. It is a common abstraction to estimate that the cost of masked non-linear operations grows quadratically in the number of shares [47]. This is traditionally motivated by the number of partial products required for executing the ISW multiplication algorithm [53]. Clearly, the main bottleneck for the efficiency of masked cipher implementations is the realization of the non-linear operations, or as commonly abstracted, the multiplications. In the following we therefore compare the performance and cost of masked multiplication algorithms and circuits. These gadgets are well-suited for a comparison as they not only consist of field multiplications but also require field addition and subtraction (which are equivalent in binary fields).

Software. First, we concentrate on software platforms. For the comparison we have chosen an STM32VLDISCOVERY board[4] with an STM32F100RB ARM

[4] https://www.st.com/en/evaluation-tools/stm32vldiscovery.html

Table 1: Cycle counts of the ISW multiplication algorithm on an ARM Cortex-M3 MCU (STM32F100RB on STM32VLDISCOVERY) for binary and prime fields with small to medium Mersenne prime exponents $3 \leq n \leq 31$.

Field	n	Number of Shares					
		1	2	3	4	5	6
$\mathbb{F}_{2^n}$	3	24	73	173	330	656	956
	5	26	75	215	388	672	968
	7	26	75	217	392	720	1032
	13	162	629	1429	2581	3933	6014
	17	210	821	1861	3349	5085	7758
	19	234	917	2077	3733	5661	8630
	31	378	1493	3373	6037	9117	13862

Field	n	Number of Shares					
		1	2	3	4	5	6
$\mathbb{F}_{2^n-1}$	3	5	20	39	104	230	382
	5	5	20	39	104	230	382
	7	5	20	39	104	230	382
	13	5	20	39	104	230	382
	17	17	82	224	468	804	960
	19	17	82	224	468	804	960
	31	19	86	230	476	820	1262

Cortex-M3 32-bit micro-controller. It provides single-cycle addition, subtraction and multiplication instructions for 32-bit operands. However, the single-cycle multiplication instruction (MUL) only produces 32-bit results and is therefore effectively a 16×16-bit multiplier for our purposes. A multi-cycle 32×32-bit multiplication instruction (UMULL) producing a 64-bit result exists, but it does not execute in constant time and is therefore not considered in this work. All our software implementations are written in C and have been compiled and analyzed using Keil MDK for ARM (MDK-Lite Version 5.36.0.0). Table 1 compares the number of cycles required to execute an ISW multiplication in constant time in both binary and Mersenne-prime fields with up to six shares (providing fifth-order security). Up to $n = 7$, the masked binary ISW multiplication is quite efficient, as the partial products can be computed using *log/alog* tables [32]. The remaining operations in the masked multiplication algorithm are XORs. For $n = 13$ and larger fields, *log/alog* tables are too big to be hard-coded as static tables in the program and to be flashed onto the device. Thus, for 13×13-bit and larger multiplications, the partial products are computed using a regular constant-time Galois field multiplication based on shift, XOR and AND operations.[5]

The masked Mersenne-prime ISW multiplication is more efficient for any number of shares on this target platform since, unlike the binary field operation, it can leverage the arithmetic multiplication instructions. The subsequent modular reduction takes another 4 cycles for small n values. For $n = 17$ and larger Mersenne primes, the multiplication result does not fit into a single 32-bit word, implying the need for multi-precision arithmetic. Using the standard Karatsuba algorithm, at most three calls to the constant-time 16-bit multiplication instruction (MUL) are needed for $17 \leq n \leq 31$. Therefore, both the multiplication and the subsequent reduction require more cycles for these sizes. Nevertheless, compared to binary field multiplications of the same size they are still significantly more efficient, by up to one order of magnitude (for $n = 31$).

[5] It is possible to compute the tables on the device when they are needed. Yet, it creates additional (memory) overheads and complicates the comparison.

This comparison obviously neglects the significant performance improvements that can be achieved for binary field operations when making use of bitslicing (first introduced in [14] to speed up DES implementations). Unfortunately this technique can not be transferred to prime fields. By contrast, using medium-sized (e.g., 31-bit) primes that leverage the data width of the target platform optimally could serve as an alternative to build fast software implementations. A more detailed comparison including such considerations falls outside the scope of this work. Our point is anyway not that prime field operations are generally cheaper in software than binary field operations, but merely that Mersenne prime field arithmetic can be implemented very efficiently on such platforms too, mainly due to the existence of optimized arithmetic hardware support. Combining with their excellent side-channel security features, it makes them promising candidates for effective and efficient masking in low noise conditions.

Hardware. For hardware implementations it is usually distinguished between Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs). In this work we mainly consider FPGAs as a suitable target platform, since similar to MCUs they already come with optimized arithmetic hardware multipliers and adders on-board. If such resources are not available, the cost of building Mersenne-prime field multiplication from combinatorial logic cells (provided by a standard cell library) is, according to our estimation, about twice as high as for binary fields of the same size (this overhead shrinks for larger n). For addition and subtraction the resource overhead is even 3-4. However, when the FPGA includes DSP slices with multipliers and adders, the utilization of soft logic (LUTs, FFs, Slices) can be significantly reduced for prime field operations, at the cost of occupying the integrated arithmetic processors.

Table 2 shows the resource utilization of ISW multiplication circuits with up to 6 shares on a Xilinx Spartan-6 FPGA (XC6SLX75-2CSG484C) for both binary and Mersenne prime fields with exponents in the range $3 \leq n \leq 31$. This target FPGA offers 132 DSP48A1 slices with one 18×18-bit multiplier each. All circuits have been implemented using Xilinx ISE Design Suite 14.7 with synthesis parameters `-keep_hierarchy` set to `yes` and `-use_dsp48` set to `Auto`. On average the masked prime field multiplications require less soft logic than the binary field equivalents, at the cost of using DSP48A1 slices. From the resource utilization figures, it becomes clear that $n = 5$ and $n = 19$ are sub-optimal choices for this target. For $n = 5$ the corresponding Mersenne prime $p = 31$ is still too small to effectively leverage the 18-bit multipliers (the synthesis tool then opted to not use a DSP slice), while its DSP-free implementation is already quite expensive. For $n = 19$ the multiplication cannot fit into an 18×18-bit multiplier, but it is also too small to effectively utilize a second instance for each multiplication, so the multiplier is extended by expensive soft logic. However, for $n = 7$, $n = 13$ and $n = 17$ (and even $n = 31$), efficient masked multiplications are found.

3.3 Larger prime ciphers

It is well-known from ECC-related research that performing cryptographic operations in larger prime fields requires expensive multi-precision arithmetic. For

Table 2: Resource consumption of the ISW multiplication algorithm on a Xilinx Spartan-6 FPGA (XC6SLX75-2CSG484C) for binary and prime fields with small to medium Mersenne prime exponents $3 \leq n \leq 31$. Note that 132 is the maximum number of available DSP48A1 slices on this target FPGA (*).

Shares	Field	n	LUTs	Slices	DSPs
1	$\mathbb{F}_{2^n}$	3	3	3	0
		5	12	6	0
		7	26	9	0
		13	81	35	0
		17	132	56	0
		19	166	71	0
		31	407	180	0
	$\mathbb{F}_{2^n-1}$	3	3	3	0
		5	34	13	0
		7	16	6	1
		13	27	12	1
		17	35	15	1
		19	116	36	1
		31	63	24	4

Shares	Field	n	LUTs	Slices	DSPs
2	$\mathbb{F}_{2^n}$	3	22	16	0
		5	64	41	0
		7	114	62	0
		13	382	157	0
		17	590	249	0
		19	742	322	0
		31	1722	802	0
	$\mathbb{F}_{2^n-1}$	3	25	19	0
		5	197	79	0
		7	121	47	4
		13	216	80	4
		17	284	99	4
		19	623	183	4
		31	520	156	16

Shares	Field	n	LUTs	Slices	DSPs
3	$\mathbb{F}_{2^n}$	3	57	46	0
		5	156	102	0
		7	273	152	0
		13	862	440	0
		17	1352	545	0
		19	1711	738	0
		31	3957	1904	0
	$\mathbb{F}_{2^n-1}$	3	66	48	0
		5	468	198	0
		7	327	123	9
		13	569	202	9
		17	737	253	9
		19	1512	444	9
		31	1335	405	36

Shares	Field	n	LUTs	Slices	DSPs
4	$\mathbb{F}_{2^n}$	3	108	84	0
		5	294	168	0
		7	500	294	0
		13	1623	610	0
		17	2414	1319	0
		19	3060	1410	0
		31	7099	335	0
	$\mathbb{F}_{2^n-1}$	3	126	85	0
		5	866	385	0
		7	600	254	16
		13	1082	386	16
		17	1428	467	16
		19	2792	816	16
		31	2527	768	64

Shares	Field	n	LUTs	Slices	DSPs
5	$\mathbb{F}_{2^n}$	3	175	128	0
		5	460	295	0
		7	821	446	0
		13	2473	1237	0
		17	3761	1826	0
		19	4808	2201	0
		31	11217	5086	0
	$\mathbb{F}_{2^n-1}$	3	205	137	0
		5	1369	575	0
		7	984	381	25
		13	1788	605	25
		17	2284	769	25
		19	4468	1301	25
		31	4095	1295	100

Shares	Field	n	LUTs	Slices	DSPs
6	$\mathbb{F}_{2^n}$	3	258	200	0
		5	672	432	0
		7	1188	714	0
		13	3657	1653	0
		17	5469	2624	0
		19	6933	3182	0
		31	16237	7110	0
	$\mathbb{F}_{2^n-1}$	3	303	211	0
		5	1966	884	0
		7	1422	612	36
		13	2625	891	36
		17	3373	1131	36
		19	6532	1894	36
		31	9254	2369	132*

this reason, most existing prime-based block ciphers are not perfectly suited to deliver the desired efficiency for low-end embedded devices. For instance, a single unmasked 129×129-bit multiplication, as required 82 times in the MiMC-129/129 block cipher [2], already costs multiple hundreds of clock cycles (without modulo reduction) on devices where single-cycle 32-bit hardware multipliers are available [38]. On a Spartan-6 FPGA a single 129×129-bit unmasked multiplication without reduction is about as expensive as 16 31×31-bit multiplications or 64 17×17-bit (or smaller) multiplications. Considering additionally that no 129-bit Mersenne prime exists, the overheads for the modulo reduction will be even more significant. The same problems arise for all prime ciphers which are either based on operations in too large fields or which require reduction modulo an implementation-unfriendly prime (since integer division might be needed). We conclude that the design space for small to medium Mersenne-prime ciphers, dedicated to efficient masked implementations, is still mostly unexplored. We next show the interest of this design space by exhibiting the advantages of a first `AES-prime` cipher over its standard version for masked implementations.

4 `AES-prime` for prime encodings

Section 2 showed that prime encodings can improve the security of the masking countermeasure in low-noise settings. Section 3 showed that the resulting operations can be implemented efficiently in hardware and software. The next step in proving the utility of these encodings is to analyze concretely-relevant computations. Yet, as mentioned in the introduction, applying prime encodings to binary ciphers is expected to lead to large performance overheads. In general, `prime` ciphers, for which the key addition is in $\mathbb{F}_p$ and the diffusion layer is linear in this field, would be better suited to this goal. As a natural starting point, we propose an `AES` variant operating in prime fields, denoted as `AES-prime`.

4.1 `AES-prime` design for $p = 2^7 - 1$

The main design guideline of `AES-prime` consists in adapting the standard `AES` design to $\mathbb{F}_p$ where p is a prime, using only additions and multiplications over the chosen prime field. The main design components are the same: the key is a vector of 16 elements of $\mathbb{F}_p$ and the state is considered as a table of 4 by 4 $\mathbb{F}_p$ elements. The round architecture is the same with the following adaptation on SubBytes, MixColumns and AddRoundKey. See [32] for details.

SubBytes. The non-linear substitution built upon the inverse function in $\mathbb{F}_{2^8}$ is replaced by a power function and the addition of a constant. The S-box is defined as $f(x) = x^e + c$ where e is the first integer such that e and $p - 1$ are co-prime, to ensure that the function mapping $x \in \mathbb{F}_p$ to x^e is a bijection, and c is the smallest positive integer such that $f(x)$ has no fix points (as in the original design). Then for $p = 2^7 - 1$ we have $e = 5$, $c = 2$ and $f(x) = x^5 + 2$.

Note that contrarily to the original `AES` nonlinear function this power map is not its own inverse. The main reason for this choice is that it allows reducing

the number of multiplications in the S-box, which is the most expensive operation to mask. Concretely, the considered x^5 mapping can be performed with three multiplications. A counterpart of this choice is that the inverse will be less efficient. For now we therefore assume that `AES-prime` will be preferably used in an inverse-free mode of operation (e.g., `CTR` [59]), leaving the investigation of S-boxes in prime fields with efficient inverses as an open problem.

MixColumns. This part of the affine layer is replaced by a 4 by 4 Maximum Distance Separable (MDS) matrix over $\mathbb{F}_p$. The reason is to guarantee a branch number of 5 as in the original `AES` design (which is optimal for this size), for diffusion properties. To choose the MDS matrix, we start from a Vandermonde matrix and perform minor modifications to decrease the number of different elements. It is beneficial to choose elements which are powers of 2 when p is a Mersenne prime, since multiplication by such a value is merely a rotation of the bits. For $p = 2^7 - 1$ it leads to the following choice:

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 16 \\ 1 & 4 & 16 & 2 \\ 1 & 16 & 2 & 4 \end{bmatrix}.$$

AddRoundKey. The bit-wise addition of the `AES` is replaced by the addition over $\mathbb{F}_p$ between the 16 elements of the round key and the state.

For completeness, we finally mention that `AES-prime` uses the key scheduling algorithm of the `AES` adapted with a prime S-box and additions modulo p. Its round constants are computed as multiples of 3 modulo p, resulting in the sequence 0x01, 0x03, 0x09, 0x1B, 0x51, 0x74, 0x5E, 0x1C, 0x54, ...

4.2 Security analysis

The proximity between the `AES-prime` and `AES` designs allows us to benefit from two decades of cryptanalysis in order to determine the security of the `AES-prime` to known attacks. For our choice of parameters, we can also lean on the security analysis of the `HADES` design strategy [48] that recently studied generalizations of SPN designs over prime fields. The main focus of [48] is on big prime sizes (where $\log_2(p)$ corresponds to the targeted security such as 128-bit) for MPC applications, but its security analysis also considers smaller sizes. For example, for $\log_2(p) = 8$ and a state of 16 words they advocate 14 rounds for a 128-bit security ($16 \times \log_2(p)$ more precisely). The main attacks exhibited against these `prime` designs are statistical attacks (mostly differential cryptanalysis), and algebraic attacks based on interpolation and Gröbner bases.

`AES-prime` has two main differences with the `HADES` ciphers. First, all rounds have a full S-box layer whereas `HADES` combines full S-box layers and partial S-box layers in order to decrease the total number of products. Second, the diffusion of `AES-prime` uses a 4 by 4 MDS matrix whereas `HADES` uses an MDS matrix on the whole state (16 by 16 in our case). We argue these differences

have a limited impact on the security analyses, and a similar number of rounds could be considered. For statistical attacks, the strategy of HADES relies on the differential and linear probabilities of an S-box and the number of active S-boxes in the full layers. The S-box of AES-prime being a power function as in HADES, we can bound the statistical probabilities using the same arguments, and we can bound the number of active S-boxes from the diffusion properties proven for the regular AES. For algebraic attacks, the strategy of HADES is based on determining the degree of the polynomials obtained after r rounds, and the number of coefficients of such polynomials. The same strategy can be applied to AES-prime. We next give more details on these attacks on AES-prime.

Statistical attacks. The most common attacks on block ciphers are linear [65] and differential cryptanalyses [16] and their variants. They consist in following how statistical biases from the S-boxes propagate along various rounds of the cipher in order to determine the key. Following the Wide Trail Strategy [31], we bound the linear and differential probability of the S-box function, and use the branch number to bound the number of active S-boxes over various rounds, in order to determine the minimal number of rounds preventing characteristics with probability higher than $2^{-\lambda}$, where λ is the security parameter. As in [48] we ensure that each characteristic has a probability smaller than $2^{-2\lambda}$ in order to avoid that a combination of various of them lead to an attack.

The differential probability of a 1-variable p-ary function f from a to b (both in $\mathbb{F}_p$) is defined as $|\{x : f(x+a) - f(x) = b\}|/|\mathbb{F}_p|$ and its linear probability relatively to a, b as $|\{x : f(x) = ax+b\}|/|\mathbb{F}_p|$.[6] Since $\mathbb{F}_p$ is a field, any polynomial of degree e has at most e roots, which gives the upper bound of $(\deg(f)-1)/p$ for the differential probabilities and $\deg(f)/p$ for the linear ones. Since in AES-prime ShiftRows is the one of the regular AES and MixColumns is based on an MDS matrix as for AES too, the number of active S-boxes in a four-round differential (or linear) trail is lower bounded by 25 (see [32], Theorem 9.5.1). Therefore, with 8 rounds the differential probabilities admit the upper bound:

$$\mathbf{p} \leq \left(\frac{\deg(f)}{p}\right)^{50}, \text{ hence for } p = 2^7 - 1 \text{ it gives } \mathbf{p} \leq \left(\frac{5}{127}\right)^{50} \approx 2^{-233}.$$

These probabilities are smaller than $2^{-2\lambda}$ and, as in [48], we add two rounds in order to guarantee that no differential attack can be set up by key guessing, which leads to a minimum of 10 rounds to avoid these attacks.

Algebraic attacks. As for the HADES framework, we consider that the main algebraic attacks threatening AES-prime are the interpolation attack [55] and attacks exploiting Gröbner bases. In both cases, the goal of the analyses are to determine a minimum number of rounds such that the polynomial representations of the cipher that an adversary can build has a too high degree of too many monomials. First, we note that for a fixed key the encryption could be studied as a function (mapping each plaintext to a ciphertext) from $\mathbb{F}_{p^{16}}$ to $\mathbb{F}_{p^{16}}$. Then, even

[6] A n-variable p-ary function is a function from $\mathbb{F}_p^n$ to $\mathbb{F}_p$.

determining the polynomial corresponding to one full S-box layer is non-trivial. Accordingly, the basis field we consider for the cryptanalyses is $\mathbb{F}_p$, and polynomials built by the adversary belong to $\mathbb{F}_p[z_0, \cdots, z_{15}]/(z_0^p - 1, \cdots, z_{15}^p - 1)$. Since the degree in one variable is at most $p-1$, the total degree is at most $16(p-1)$ and there are p^{16} monomials. The S-box used in `AES-prime` being a power function x^e added to a constant, the total degree will therefore increase as e^r for the first r rounds, until reaching the maximum of $16(p-1)$. For the number of monomials, most of the monomials will be present in each polynomial after a few rounds (most since even for a random polynomial each monomial is present with probability $(p-1)/p$). Indeed, after $\lceil \log_e(p) \rceil$ rounds, all the univariate monomials (z_i^j) are obtained. Two more rounds ensure to have sums with each one of the 16 variables (at some power) as input of the S-box' f function (defined in Section 4.1). As a result, $\lceil \log_e(p) \rceil$ more rounds are sufficient to obtain all monomials. Overall, we obtain dense polynomials after $2\lceil \log_e(p) \rceil + 2$ rounds, which corresponds to 10 rounds for the chosen $p = 2^7 - 1$.

For the interpolation attack, the attacker aims at interpolating a polynomial from $\mathbb{F}_p^{16}$ to $\mathbb{F}_p$ corresponding to the encryption over all minus one rounds using known plaintext/ciphertext pairs. If such a polynomial has a low degree or few monomials, the adversary can guess the key of the final round, decrypt the ciphertexts, interpolate the polynomial corresponding to all minus one rounds and confirm it with an extra plaintext/ciphertext pair. The data cost is well approximated by the number of plaintext/ciphertext pairs necessary to build such a polynomial. Following the strategy of `HADES`, we consider that such an attack cannot succeed when the number of monomials in the cipher polynomial is equal to the full code book, since it corresponds to $p^{16} \approx 2^\lambda$ monomials which already meets the targeted security. Accordingly, we count $2\lceil \log_e(p) \rceil + 3$ rounds to rule out the interpolation attack, 11 for $p = 2^7 - 1$. Due to the proximity of design between `SHARK` [78] and `AES-prime`, we also consider the analysis of the first interpolation attack [55] on this block cipher. The principle of this attack is that even if the S-box corresponds to a function of high degree (maximal in the case of the inverse as in `SHARK`), it can be attacked more easily in another representation (e.g., as a fraction of low-degree polynomials). In this case, the complexity of the attack comes from the number of S-boxes rather than their size or degree. The complexity of the best attack is then at least $2(t^{r-3})^t$, where t is the number of S-boxes and r the number of rounds. For the `AES-prime` it corresponds to $2 \cdot 16^{16(r-3)}$, hence at least 5 rounds for $p = 2^7 - 1$.

For Gröbner bases attacks, the attacker aims at solving a system of multivariate polynomials over $\mathbb{F}_p$ in the key elements obtained with sufficient plaintext/ciphertext pairs. Determining the (tight) complexity of these attacks is impossible. Hence the security of ciphers against these attacks is usually based on the infeasibility of computing the Gröbner basis in degree reverse lexical order. We follow this strategy also used in `HADES`. Since the design differences (e.g., MDS matrix on partial or full state, full or partial S-boxes layers) have no influence on the final complexity, we respect the bound of at least $2 + \lceil \log_e(p)/2 \rceil + \lceil \log_e(16) \rceil$ rounds. For $p = 2^7 - 1$ it leads to a minimum of 6 rounds.

Number or rounds. Based on the complexity of the different attacks considered, 11 rounds would be sufficient for the targeted security of $\lfloor 16 \log(2^7 - 1) \rfloor = 111$ bits for this value of p, which is coherent with the number of rounds in the `AES`. Since various improvement of the considered attacks, or attacks of the same families, are possible we take a more conservative approach and follow the estimations of `HADES`. Accordingly, we advocate the use of 14 rounds.

Before moving to the experimental validation of our findings, we re-insist that the proposed instance of the `AES-prime` cipher with $p = 2^7 - 1$ is only aimed to confirm the interesting design space that prime ciphers open. In particular, our following conclusions only require that `AES`-like ciphers operating in binary and prime fields of similar sizes require similar number of rounds (i.e., differ by factors that are covered by the physical security gains that the `AES-prime` provides). Besides, we note that this design is scalable. For example, if 128 bits of security or more are required, a 13-bit variant with $p = 2^{13} - 1$ can be considered. The S-box could then be based on $f(x) = x^{11} + 3$ while keeping the same MDS matrix as the 7-bit instance for MixColumns, and the same number of rounds.

5 Experimental validation

In this last section, we finally consider the practical impact of prime encodings on the security of masked block cipher implementations in software and hardware. For this purpose we implement masked field multiplications in $\mathbb{F}_{2^7-1}$ based on the ISW multiplication algorithm [53] and construct probing secure implementations of the `AES-prime` S-box, i.e., $x^5 + 2$. We refrain from any comparison to masked versions of the standard AES S-box here, which is based on Galois field inversion in $\mathbb{F}_{2^8}$. The different bit lengths of inputs processed, the different number and size of field multiplications required and the various known implementation strategies for the standard AES S-box are among the reasons why any such comparison would depend a lot on the ad-hoc choices made along the way and indeed feel like comparing apples to oranges. Thus, we chose to compare the identical operations, i.e., multiplication and $f(x) = x^5 + 2$, in the corresponding fields $\mathbb{F}_{2^7-1}$ and $\mathbb{F}_{2^7}$. We stress that the following results are not meant as an efficiency comparison (which would favor binary fields for this choice of S-box since the squaring operation is linear in $\mathbb{F}_{2^7}$), but merely as a comparison of their side-channel leakage. For our software case study, we evaluate the security of these implementations for an increasing number of shares (up to 6) against a profiled SASCA attack (similar to the one considered in [20]). In this setting the adversary is given full profiling access to the device to characterize its leakage behavior and build optimal models for the attack. Furthermore, the chosen 32-bit MCU (specified next) has shown a low natural noise level when measuring its power consumption. As a second case study, we evaluate the leakage reduction offered by prime field operations in hardware compared to the equivalent binary field operations using a detection-based leakage assessment [81].

5.1 Target devices and experimental setups

For the software-based investigations, we have targeted the same device as already described in Section 3, namely an STM32VLDISCOVERY board with an embedded STM32F100RB ARM Cortex-M3 32-bit micro-controller. Both the discovery board and the MCU are identical to the Cortex-M3 experiments presented in [20], where bitslice masked implementations have been analyzed. Also similar to [20], we have conducted a standard modification of the board for power measurements, namely carefully removing the decoupling capacitors in the power grid of the target chip to obtain improved results. The Cortex-M3 has been operated at 8 MHz throughout our experiments.

For the hardware-based investigation we have evaluated the masked implementations on a SAKURA-G FPGA board,[7] which employs two Xilinx Spartan-6 FPGAs: one as a target and one as a control unit. The target FPGA is the exact Spartan-6 device we have used for the estimation of the resource consumption of masked field multiplications in Section 3. We have operated the target implementations at 6 MHz. Low frequencies were selected since we are interested in conducting our comparisons at minimal noise levels.

In both cases, we have used the same measurement equipment and settings. In detail, we have placed a CT-1 current probe from Tektronix with a bandwidth up to 1 GHz in the power supply path of the target FPGA and acquired the measurements with a PicoScope 5244D digital sampling oscilloscope. The sampling rate was set to 250 MS/s and the vertical resolution was 12 bit.

5.2 Software case study

The goal of the software-based case study is to validate whether moving to encodings and operations in prime fields actually leads to concrete security improvements compared to standard Boolean masking in real-world experiments. To answer this question, we mount horizontal attacks against the `AES-prime` S-box for two different encodings and two implementation strategies realized on a small 32-bit micro-controller (known to provide limited natural noise).

Methodology. We implemented the S-box as a sequence of three ISW multiplications , as described in Section 4. It gives rise to intermediate computations x, x^2, x^4, x^5. We added refreshing gadgets according to the trivial composition strategy of [47,26]. We follow the attack of Bronchain and Standaert from CHES 2021 [20] to efficiently leverage the horizontal leakages and extend it to operations in prime fields. Concretely, this analytical attack targets all the encodings appearing in the multiplication chain, which has the significant advantage that the resulting factor graph does not have cycles and is guaranteed to converge.[8]

[7] http://satoh.cs.uec.ac.jp/SAKURA/index.html

[8] We also tested SASCA with the full factor graph, using the heuristic of running the BP algorithm for a number of steps corresponding to twice the diameter of the factor graph. This attack variant did not lead to significant improvements.

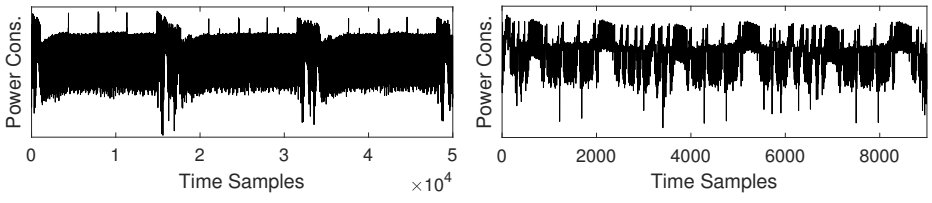
Table 3: Concrete cycle counts (left) and resource utilization figures (right) of the software and hardware implementations measured in this section. All values are for constant time implementations and exclude randomness generation.

	Field Arith.		log/alog	
d	$\mathbb{F}_{2^n}$	$\mathbb{F}_{2^n-1}$	$\mathbb{F}_{2^n}$	$\mathbb{F}_{2^n-1}$
2	1321	189	232	282
3	2902	334	448	535
4	5213	600	800	912
5	8255	1125	1340	1581
6	12038	1692	1988	2283

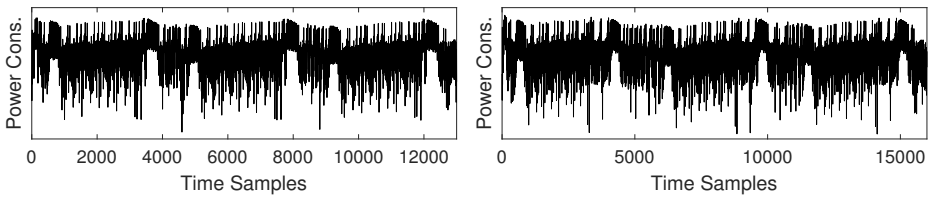
	Binary Field $\mathbb{F}_{2^n}$			Prime Field $\mathbb{F}_{2^n-1}$		
d	LUTs	Slic.	DSPs	LUTs	Slic.	DSPs
2	26	15	0	20	11	1
3	126	77	0	131	70	4
4	285	161	0	348	160	9
5	539	293	0	710	306	16
6	848	486	0	1096	515	25

The procedure of the attack is as follows. We assume that the adversary can profile the leakage of the device while even knowing the random values used during the profiling phase. This is a standard assumption when trying to evaluate the security in the worst-case scenario for the designer by considering a very strong attacker model [20]. The adversary uses the profiles subsequently to target each share and intermediate value separately and finally combines all acquired information to determine the underlying secrets. For this purpose, we first select representative Points-of-Interest (POIs) in the trace for each relevant intermediate value thanks to the standard Signal-to-Noise Ratio (SNR) metric [60]. The leakage distribution of each target intermediate value is then profiled thanks to a pooled template attack [29] after a dimensionality reduction step using Linear Discriminant Analysis (LDA) [83]. As a result, we obtain leakage models for all the target intermediate values of the factor graph and use them in place of the *a priori* (Hamming weight and LSB) ones of Section 2. These multivariate models can be superior to the previous HW and LSB models and may enable attacks with very few traces. Since we are interested in the practical interest of prime encodings, we also switch from (easier to estimate) information theoretic metrics to a security metric, namely the Guessing Entropy (GE) [84] that captures the average key rank and which we estimated based on $1,000$ different attacks.

Experimental results. We have implemented the masked computation of x^5+2 in both fields $\mathbb{F}_{2^7}$ and $\mathbb{F}_{2^7-1}$ using two different implementation strategies. First, using regular field arithmetic for the multiplication, which includes shift, XOR and AND operations for the binary field and single-cycle multiplication and addition instructions for the prime field. However, since these approaches show a vastly different leakage pattern, we have additionally realized a table-lookup based implementation using *log/alog* tables for both fields. The table-lookup based implementations are realized using very similar sequences of instructions and are therefore perfectly suited for a fair comparison. It is important to mention that all implementations work in constant time. For the *log/alog* table implementations we have only considered traces for the attack where the inputs to the lookup-based field multiplication are non-zero. Despite the fact that all



(a) $x^5 + 2$ in $\mathbb{F}_{2^7}$, (field arithmetic) (b) $x^5 + 2$ in $\mathbb{F}_{2^7-1}$, (field arithmetic)

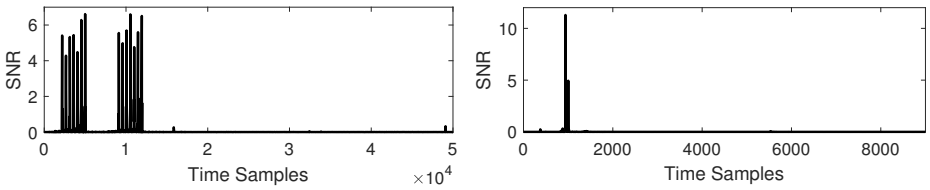


(c) $x^5 + 2$ in $\mathbb{F}_{2^7}$, (log/alog tables) (d) $x^5 + 2$ in $\mathbb{F}_{2^7-1}$, (log/alog tables)

Fig. 5: Illustrative Cortex-M3 sample traces of a first-order masked implementation computing $x^5 + 2$ in binary field (left) and prime field (right) using regular field arithmetic operations (top) and log/alog tables (bottom).

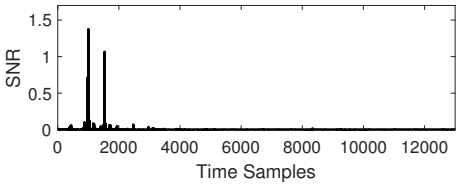
our implementations execute in constant time (for any input), there is still an operation dependency in the case that one or both multiplication inputs are zero (see [32] for a description of the lookup function). This dependency, which is inherent to all *log/alog* table based implementations, allows to trivially identify all zero-inputs with probability one in the traces. In order to avoid this special case in our comparison we have only considered non-zero inputs for *both* binary and prime fields. For our purposes this simple workaround was acceptable, for real implementations we recommend more prudent strategies such as presented in [80]. Sample traces of the acquired measurements from the Cortex-M3 for the four different implementations are shown in Figure 5 for the case of 2 shares. We have repeated those measurements for 3, 4, 5 and 6 shares for each of the four cases. For completeness, we provide the concrete cycle counts and resource utilization figures of the analyzed implementations in Table 3. As detailed before, the cycle counts required for performing three consecutive masked multiplications are compared in the software case (which is clearly not the most efficient manner to compute x^5, especially in binary fields). Please note that log/alog table based prime implementations require slightly more cycles than the equivalent Boolean implementations, since only the partial multiplications are performed via table lookup, while additions are still performed using regular arithmetic (requiring a reduction in the prime case). Figure 6 for example shows the Signal-to-Noise Ratio (SNR) for one input share per implementation. As expected, the leakage patterns and magnitudes are different between binary and prime field computation when considering the regular field arithmetic implementations. However, they are strikingly similar for the table-based implementations.

In order to perform the attacks we have built profiles using 50,000 traces each, selected a maximum of 200 POIs per variable and set the number of di-

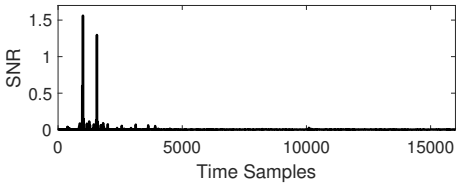


(a) $x^5 + 2$ in $\mathbb{F}_{2^7}$, (field arithmetic)

(b) $x^5 + 2$ in $\mathbb{F}_{2^7-1}$, (field arithmetic)



(c) $x^5 + 2$ in $\mathbb{F}_{2^7}$, (log/alog tables)



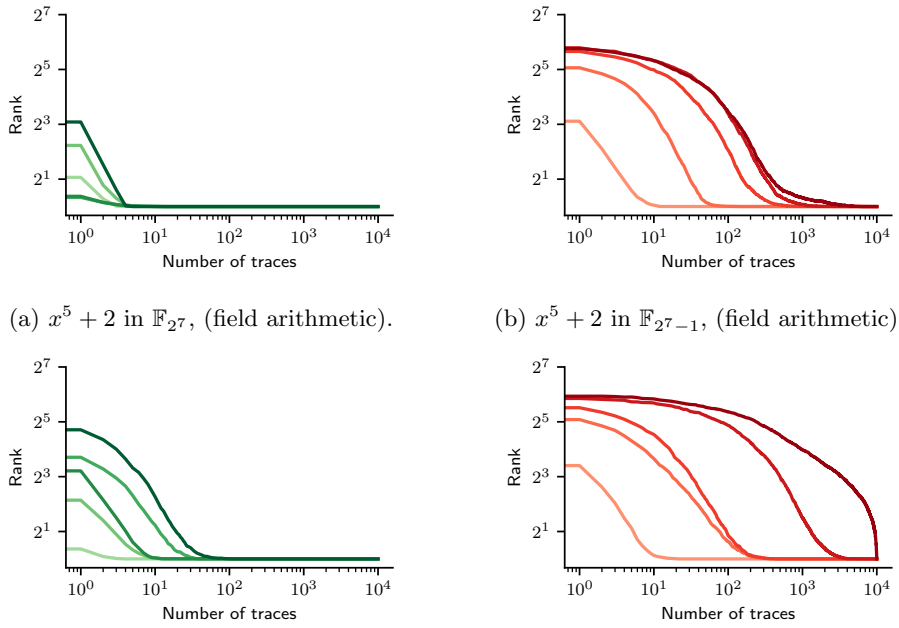
(d) $x^5 + 2$ in $\mathbb{F}_{2^7-1}$, (log/alog tables)

Fig. 6: Signal-to-Noise Ratio (SNR) of input share 0 for the traces in Figure 5.

mensions after LDA projection to 2. The results are depicted in Figure 7. As expected for a low-noise device, Boolean masking (in $\mathbb{F}_{2^7}$) with multiplication based on field arithmetic (Figure 7a) leads to trivial attacks in less than 10 traces against all implementations with up to 6 shares using the profiled models. By contrast, the corresponding SASCA against the `AES-prime` S-box (Figure 7b) requires significantly more observations to succeed. Attacking the 6-share implementation requires around $4,000$ traces. Yet, the differences between Figure 7b and Figure 7a could potentially be influenced by the way the field multiplication is implemented. Therefore, we repeated the attacks for the table-based implementations, where the leakage per share has been shown to be equivalent. The result is depicted in Figures 7c and 7d. It can be observed that the attacks require more traces to succeed for both fields due to the lower SNR (c.f. Figure 6). Yet, 50-60 traces still suffice to retrieve the key of the Boolean masked computation with 6 shares, while about $10,000$ traces are required for the equivalent prime field masked S-box. These results confirm the interest of prime encodings. In particular, they show a significant security benefit when increasing the number of shares for prime field masking even in this challenging low-noise software context, which is a major advantage over Boolean masking.

5.3 Hardware case study

In order to investigate whether similarly impressive security improvements can be achieved for the (naturally more noisy) case of parallel hardware implementations, we have conducted a second case study. In this experiment, we implemented the ISW multiplication algorithm in hardware and, as for the previous case study, compare the security provided by Boolean (in $\mathbb{F}_{2^7}$) and prime field (in $\mathbb{F}_{2^7-1}$) masking. When implemented in two cycles (and synchronizing the outputs with a register [69]), the ISW multiplication algorithm leads to a ro-



(a) $x^5 + 2$ in $\mathbb{F}_{2^7}$, (field arithmetic).

(b) $x^5 + 2$ in $\mathbb{F}_{2^7-1}$, (field arithmetic).

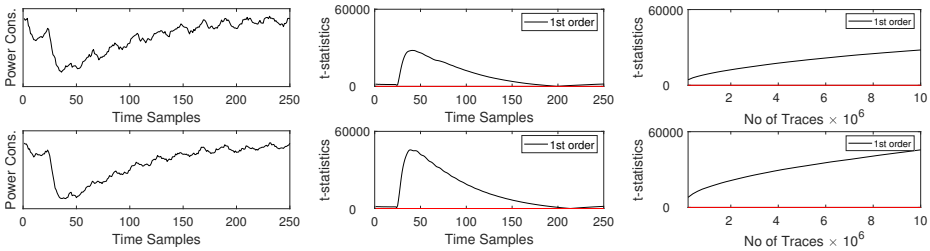
(c) $x^5 + 2$ in $\mathbb{F}_{2^7}$, (log/alog tables).

(d) $x^5 + 2$ in $\mathbb{F}_{2^7-1}$, (log/alog tables).

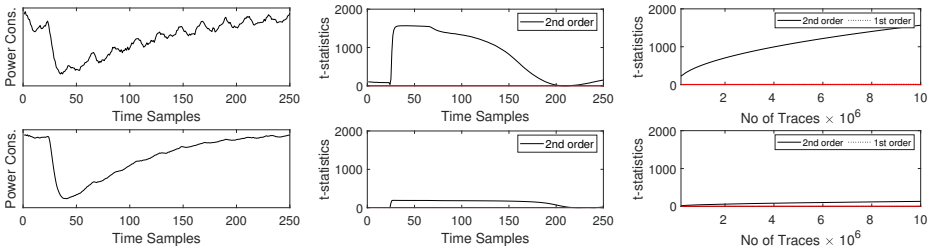
Fig. 7: Guessing Entropy of SASCA against a software implementation of the `AES-prime` S-box (top) and a binary variant (bottom) for 2 to 6 shares.

bust probing secure and composable circuit gadget [44]. This is crucial to avoid a reduction of the statistical security order due to the presence of physical defaults in the hardware such as glitches. We implemented the ISW multiplication in both considered fields for up to five shares in a fully pipelined manner. A comparison of their resource utilization on the Spartan-6 FPGA is given in Table 3. As detailed in Section 3, the prime field multiplications are supported by DSP slices instead of pure soft logic implementations. We only execute and measure a single masked multiplication gadget to obtain relatively low noise levels (aside from the parallelism inside the gadget itself). Then we perform a Test Vector Leakage Assessment (TVLA)-based analysis [81] to verify the security order provided by the circuits and analyze the number of traces required to exceed the detection threshold. We chose to perform this analysis using two fixed classes (0x00 · 0x00 vs. 0x7E · 0x7E) to minimize the number of measurements required to detect data-dependent information, as suggested in [39]. Our results are depicted in Figure 8. Table 4 lists the required amounts of traces to pass the detection threshold ($t > 4.5$) in the respective TVLA procedures.

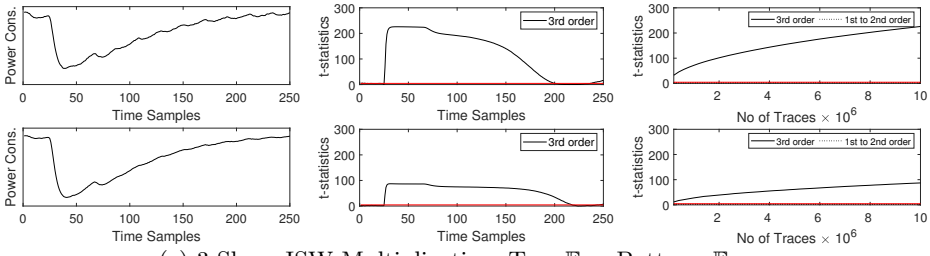
For the unmasked case, the difference is insignificant. However, when increasing the number of shares, the amount of traces required to confidently detect leakage grows much more quickly for the prime field multiplication than for the binary field one. Concretely, the 4- and 5-share masked multiplications in



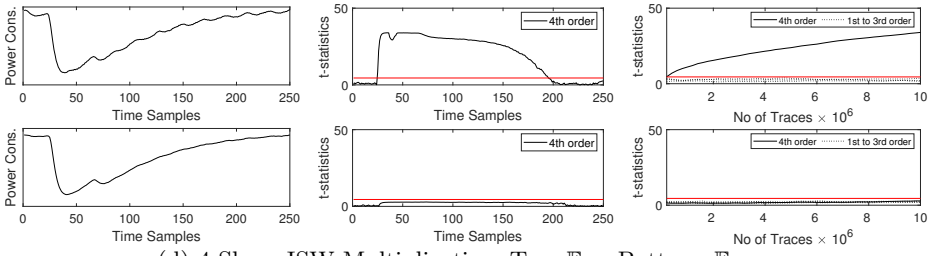
(a) Unmasked Multiplication, Top: $\mathbb{F}_{2^7}$, Bottom: $\mathbb{F}_{2^7-1}$



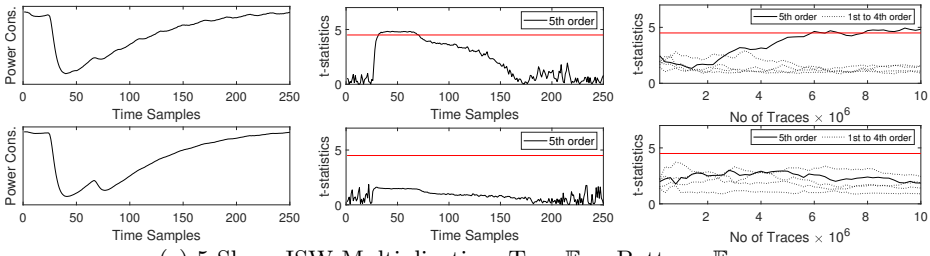
(b) 2-Share ISW Multiplication, Top: $\mathbb{F}_{2^7}$, Bottom: $\mathbb{F}_{2^7-1}$



(c) 3-Share ISW Multiplication, Top: $\mathbb{F}_{2^7}$, Bottom: $\mathbb{F}_{2^7-1}$



(d) 4-Share ISW Multiplication, Top: $\mathbb{F}_{2^7}$, Bottom: $\mathbb{F}_{2^7-1}$



(e) 5-Share ISW Multiplication, Top: $\mathbb{F}_{2^7}$, Bottom: $\mathbb{F}_{2^7-1}$

Fig. 8: TVLA-based comparison of ISW multiplications. Left to right: sample trace, TVLA over points, TVLA over traces.

Table 4: Required numbers of traces to pass the detection threshold.

Field $\backslash d$	1	2	3	4	5
$\mathbb{F}_{2^n}$	< 10	100	20,300	198,000	5,870,000
$\mathbb{F}_{2^n-1}$	< 10	3,700	128,100	$> 10,000,000$	$> 10,000,000$

$\mathbb{F}_{2^7}$ still show confidently detectable leakage after approximately 198,000 and 5,870,000 traces respectively. The analysis of the corresponding $\mathbb{F}_{2^7-1}$ multiplications on the other hand can not find enough input-dependent information in 10,000,000 traces to distinguish the two fixed classes with a confidence above the threshold. For the 4-share case this means that even a 50 times larger number of observations is insufficient to achieve the same detection result.

We note that in the 3-share case, the relative difference between the metrics for binary and prime fields appears to be smaller. Yet, we believe that it does not reflect a smaller relative difference between the practical security levels provided by these two implementations, but is instead owed to known shortcomings of the TVLA procedure [87]. Indeed, there are factors beyond the security of the implementation that influence the magnitude of the statistic and also the number of traces required to reach a certain magnitude in TVLA (e.g., we verified that the relative gap in the 3-share case is larger for different choices of the input classes). So as usual with leakage detection, these results should be used as first hints towards the significantly improved security that prime field masking offers, especially given the overwhelming differences in the higher order cases. But they are not a directly suitable way to conclude about a security level expressed in terms of number of traces to recover the key. We leave such an advanced analysis of worst-case attacks as an interesting scope for further research.

6 Conclusions

The results in this paper show that masking with prime encodings can lead to major security improvements in the practically-relevant case of devices with low noise. Evaluations in software and hardware show security improvements by orders of magnitude over Boolean masking (for targets of equivalent sizes). We hope these results open the way to a better understanding of masking in this challenging context. We also believe they lead to important open problems.

A first direction is to further improve security by decreasing the side-channel signal. One option for this purpose is to work with larger p values, which will also raise evaluation challenges, as it implies the need to profile larger intermediate computations (which is expensive with current tools). Another one is to leverage algorithmic noise, either by using random values that are larger than p in the ISW multiplications, or by performing all the computations modulo $p \cdot N$ (where N would be an algorithmic noise parameter). A second direction is to further study the physical cryptanalysis of prime encodings. For example, investigating algebraic attacks could be relevant (although also raising challenges, as these

attacks generally have a limited noise tolerance that may not be adapted to masking) [77,22,73]. As mentioned in the introduction, assessing the interest of prime encodings in the context of static leakage is another interesting question. And of course, the design of ciphers that are even better suited to masking in prime fields than the `AES-prime`, and their comparison with optimized bitslice ciphers implemented with Boolean masking is an important long-term goal.

Acknowledgments. François-Xavier Standaert is senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded by the EU through the ERC project SWORD (724725).

References

1. Albrecht, M.R., Grassi, L., Perrin, L., Ramacher, S., Rechberger, C., Rotaru, D., Roy, A., Schofnegger, M.: Feistel structures for MPC, and more. In: ESORICS 2019 (2019). https://doi.org/10.1007/978-3-030-29962-0_8
2. Albrecht, M.R., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: ASIACRYPT 2016 (2016). https://doi.org/10.1007/978-3-662-53887-6_7
3. Aldous, D., Diaconis, P.: Shuffling cards and stopping times. The American Mathematical Monthly **93**(5), 333–348 (1986). https://doi.org/10.1080/00029890.1986.11971821
4. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. ToSC **2020**(3). https://doi.org/10.13154/tosc.v2020.i3.1-45
5. Andrychowicz, M., Dziembowski, S., Faust, S.: Circuit compilers with $O(1/\log(n))$ leakage rate. In: EUROCRYPT 2016 [43], pp. 586–615. https://doi.org/10.1007/978-3-662-49896-5_21
6. Bajard, J.C., Duquesne, S.: Montgomery-friendly primes and applications to cryptography. JCEN **11**(4) (2021). https://doi.org/10.1007/s13389-021-00260-z
7. Balasch, J., Faust, S., Gierlichs, B., Paglialonga, C., Standaert, F.X.: Consolidating inner product masking. In: ASIACRYPT 2017 (2017). https://doi.org/10.1007/978-3-319-70694-8_25
8. Balasch, J., Gierlichs, B., Grosso, V., Reparaz, O., Standaert, F.: On the cost of lazy engineering for masked software implementations. In: CARDIS (2014). https://doi.org/10.1007/978-3-319-16763-3_5
9. Barthe, G., Belaïd, S., Cassiers, G., Fouque, P.A., Grégoire, B., Standaert, F.X.: maskVerif: Automated verification of higher-order masking in presence of physical defaults. In: ESORICS 2019 (2019). https://doi.org/10.1007/978-3-030-29959-0_15
10. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y.: Verified proofs of higher-order masking. In: Oswald and Fischlin [74]. https://doi.org/10.1007/978-3-662-46800-5_18
11. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: CCS 2016 (Oct 2016). https://doi.org/10.1145/2976749.2978427
12. Battistello, A., Coron, J.S., Prouff, E., Zeitoun, R.: Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In: CHES 2016 (Aug 2016). https://doi.org/10.1007/978-3-662-53140-2_2

13. Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Randomness complexity of private circuits for multiplication. In: EUROCRYPT 2016 [43]. https://doi.org/10.1007/978-3-662-49896-5_22
14. Biham, E.: A fast new DES implementation in software. In: FSE 1997 [15]. https://doi.org/10.1007/BFb0052352
15. Biham, E. (ed.): FSE'97, vol. 1267 (Jan 1997)
16. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: CRYPTO'90 (1991). https://doi.org/10.1007/3-540-38424-3_1
17. Bloem, R., Groß, H., Iusupov, R., Könighofer, B., Mangard, S., Winter, J.: Formal verification of masked hardware implementations in the presence of glitches. In: Nielsen and Rijmen [71]. https://doi.org/10.1007/978-3-319-78375-8_11
18. Bos, J.W., Costello, C., Hisil, H., Lauter, K.: Fast cryptography in genus 2. In: EUROCRYPT 2013 [42]. https://doi.org/10.1007/978-3-642-38348-9_12
19. Bos, J.W., Costello, C., Longa, P., Naehrig, M.: Selecting elliptic curves for cryptography: an efficiency and security analysis. JCEN **6**(4) (2016). https://doi.org/10.1007/s13389-015-0097-y
20. Bronchain, O., Standaert, F.X.: Breaking masked implementations with many shares on 32-bit software platforms. TCHES **2021**(3). https://doi.org/10.46586/tches.v2021.i3.202-234
21. Buhan, I., Schneider, T. (eds.): CARDIS 2022 (2023). https://doi.org/10.1007/978-3-031-25319-5
22. Carlet, C., Faugère, J.C., Goyet, C., Renault, G.: Analysis of the algebraic side channel attack. JCEN **2**(1). https://doi.org/10.1007/s13389-012-0028-0
23. Cassiers, G., Grégoire, B., Levi, I., Standaert, F.: Hardware private circuits: From trivial composition to full verification. IEEE Trans. Computers **70**(10) (2021). https://doi.org/10.1109/TC.2020.3022979
24. Cassiers, G., Standaert, F.X.: Provably secure hardware masking in the transition- and glitch-robust probing model: Better safe than sorry. TCHES **2021**(2). https://doi.org/10.46586/tches.v2021.i2.136-158
25. Cassiers, G., Standaert, F.X.: Towards globally optimized masking: From low randomness to low noise rate. TCHES **2019**(2). https://doi.org/10.13154/tches.v2019.i2.162-198
26. Cassiers, G., Standaert, F.: Trivially and efficiently composing masked gadgets with probe isolating non-interference. TIFS **15** (2020). https://doi.org/10.1109/TIFS.2020.2971153
27. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: CRYPTO'99 (1999). https://doi.org/10.1007/3-540-48405-1_26
28. Cho, J., Ha, J., Kim, S., Lee, B., Lee, J., Lee, J., Moon, D., Yoon, H.: Transciphering framework for approximate homomorphic encryption. In: ASIACRYPT 2021 (2021). https://doi.org/10.1007/978-3-030-92078-4_22
29. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: CARDIS (2013). https://doi.org/10.1007/978-3-319-08302-5_17
30. Coron, J.S., Giraud, C., Prouff, E., Renner, S., Rivain, M., Vadnala, P.K.: Conversion of security proofs from one leakage model to another: A new issue. In: COSADE 2012 (2012). https://doi.org/10.1007/978-3-642-29912-4_6
31. Daemen, J., Rijmen, V.: The wide trail design strategy. In: IMACC (2001). https://doi.org/10.1007/3-540-45325-3_20
32. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002). https://doi.org/10.1007/978-3-662-04722-4

33. de Chérisey, E., Guilley, S., Rioul, O., Piantanida, P.: Best information is most successful. TCHES **2019**(2). https://doi.org/10.13154/tches.v2019.i2.49-79
34. Dobraunig, C., Grassi, L., Guinet, A., Kuijsters, D.: Ciminion: Symmetric encryption based on Toffoli-gates over large finite fields. In: EUROCRYPT 2021 (2021). https://doi.org/10.1007/978-3-030-77886-6_1
35. Dobraunig, C., Grassi, L., Helminger, L., Rechberger, C., Schofnegger, M., Walch, R.: Pasta: A case for hybrid homomorphic encryption. Cryptology ePrint Archive, Report 2021/731 (2021), https://eprint.iacr.org/2021/731
36. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: EUROCRYPT 2014 (2014). https://doi.org/10.1007/978-3-642-55220-5_24
37. Duc, A., Faust, S., Standaert, F.X.: Making masking security proofs concrete - or how to evaluate the security of any leaking device. In: Oswald and Fischlin [74]. https://doi.org/10.1007/978-3-662-46800-5_16
38. Düll, M., Haase, B., Hinterwälder, G., Hutter, M., Paar, C., Sánchez, A.H., Schwabe, P.: High-speed curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers. Des. Codes Cryptogr. **77**(2-3) (2015). https://doi.org/10.1007/s10623-015-0087-1
39. Durvaux, F., Standaert, F.X.: From improved leakage detection to the detection of points of interests in leakage traces. In: EUROCRYPT 2016 (2016). https://doi.org/10.1007/978-3-662-49890-3_10
40. Dziembowski, S., Faust, S.: Leakage-resilient circuits without computational assumptions. In: TCC 2012 (2012). https://doi.org/10.1007/978-3-642-28914-9_13
41. Dziembowski, S., Faust, S., Skórski, M.: Optimal amplification of noisy leakages. In: TCC 2016-A (2016). https://doi.org/10.1007/978-3-662-49099-0_11
42. EUROCRYPT 2013 (May 2013)
43. EUROCRYPT 2016 (2016)
44. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.X.: Composable masking schemes in the presence of physical defaults & the robust probing model. TCHES **2018**(3). https://doi.org/10.13154/tches.v2018.i3.89-120
45. Fumaroli, G., Martinelli, A., Prouff, E., Rivain, M.: Affine masking against higher-order side channel analysis. In: SAC 2010 (2011). https://doi.org/10.1007/978-3-642-19574-7_18
46. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: CHES'99 (1999). https://doi.org/10.1007/3-540-48059-5_15
47. Goudarzi, D., Rivain, M.: How fast can higher-order masking be in software? In: EUROCRYPT 2017 (2017). https://doi.org/10.1007/978-3-319-56620-7_20
48. Grassi, L., Lüftenegger, R., Rechberger, C., Rotaru, D., Schofnegger, M.: On a generalization of substitution-permutation networks: The HADES design strategy. In: EUROCRYPT 2020 (2020). https://doi.org/10.1007/978-3-030-45724-2_23
49. Grégoire, B., Papagiannopoulos, K., Schwabe, P., Stoffelen, K.: Vectorizing higher-order masking. In: COSADE 2018 (2018). https://doi.org/10.1007/978-3-319-89641-0_2
50. Groß, H., Mangard, S., Korak, T.: An efficient side-channel protected AES implementation with arbitrary protection order. In: CT-RSA 2017 (2017). https://doi.org/10.1007/978-3-319-52153-4_6
51. Grosso, V., Standaert, F.X.: Masking proofs are tight and how to exploit it in security evaluations. In: Nielsen and Rijmen [71]. https://doi.org/10.1007/978-3-319-78375-8_13

52. Hamburg, M.: Fast and compact elliptic-curve cryptography. IACR Cryptol. ePrint Arch. p. 309 (2012), http://eprint.iacr.org/2012/309
53. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: CRYPTO 2003 (2003). https://doi.org/10.1007/978-3-540-45146-4_27
54. Ito, A., Ueno, R., Homma, N.: On the success rate of side-channel attacks on masked implementations: Information-theoretical bounds and their practical usage. In: CCS 2022 (2022). https://doi.org/10.1145/3548606.3560579
55. Jakobsen, T., Knudsen, L.R.: The interpolation attack on block ciphers. In: Biham [15]. https://doi.org/10.1007/BFb0052332
56. Kloss, B.M.: Probability distributions on bicompact topological groups. Theory of Probability & Its Applications **4**(3) (1959). https://doi.org/10.1007/s10623-015-0087-1
57. Knichel, D., Sasdrich, P., Moradi, A.: SILVER - statistical independence and leakage verification. In: ASIACRYPT 2020 (2020). https://doi.org/10.1007/978-3-030-64837-4_26
58. Krachenfels, T., Ganji, F., Moradi, A., Tajik, S., Seifert, J.P.: Real-world snapshots vs. theory: Questioning the t-probing security model. In: 2021 Symposium on Security and Privacy (May 2021). https://doi.org/10.1109/SP40001.2021.00029
59. Lipmaa, H., Rogaway, P., Wagner, D.: Ctr-mode encryption. In: First NIST Workshop on Modes of Operation. vol. 39. Citeseer. MD (2000)
60. Mangard, S.: Hardware countermeasures against DPA – A statistical analysis of their effectiveness. In: CT-RSA 2004 (2004). https://doi.org/10.1007/978-3-540-24660-2_18
61. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: CT-RSA 2005 (2005). https://doi.org/10.1007/978-3-540-30574-3_24
62. Mangard, S., Pramstaller, N., Oswald, E.: Successfully attacking masked AES hardware implementations. In: CHES 2005 (Aug / Sep 2005). https://doi.org/10.1007/11545262_12
63. Masure, L., Rioul, O., Standaert, F.: A nearly tight proof of Duc et al.'s conjectured security bound for masked implementations. In: Buhan and Schneider [21]. https://doi.org/10.1007/978-3-031-25319-5_4
64. Masure, L., Cristiani, V., Lecomte, M., Standaert, F.X.: Don't learn what you already know: Scheme-aware modeling for profiling side-channel analysis against masking. TCHES **2023**(1). https://doi.org/10.46586/tches.v2023.i1.32-59
65. Matsui, M.: Linear cryptanalysis method for DES cipher. In: EUROCRYPT'93 (1994). https://doi.org/10.1007/3-540-48285-7_33
66. Moos, T.: Static power SCA of sub-100nm CMOS ASICs. TCHES **2019**(3). https://doi.org/10.13154/tches.v2019.i3.202-232
67. Moos, T., Moradi, A.: Countermeasures against static power attacks. TCHES **2021**(3). https://doi.org/10.46586/tches.v2021.i3.780-805
68. Moos, T., Moradi, A., Richter, B.: Static power side-channel analysis of a threshold implementation prototype chip. In: DATE. pp. 1324–1329. IEEE (2017). https://doi.org/10.23919/DATE.2017.7927198
69. Moos, T., Moradi, A., Schneider, T., Standaert, F.X.: Glitch-resistant masking revisited. TCHES **2019**(2), 256–292 (2019). https://doi.org/10.13154/tches.v2019.i2.256-292
70. Moradi, A.: Side-channel leakage through static power - should we care about in practice? In: CHES 2014 (2014). https://doi.org/10.1007/978-3-662-44709-3_31

71. Nielsen, J.B., Rijmen, V. (eds.): EUROCRYPT 2018, Part II, vol. 10821 (Apr / May 2018)
72. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. Journal of Cryptology **24**(2) (2011). https://doi.org/10.1007/s00145-010-9085-7
73. Oren, Y., Renauld, M., Standaert, F.X., Wool, A.: Algebraic side-channel attacks beyond the hamming weight leakage model. In: CHES 2012 (2012). https://doi.org/10.1007/978-3-642-33027-8_9
74. Oswald, E., Fischlin, M. (eds.): EUROCRYPT 2015, Part I, vol. 9056 (Apr 2015)
75. Pozo, S.M.D., Standaert, F., Kamel, D., Moradi, A.: Side-channel attacks from static power: when should we care? In: DATE. ACM (2015)
76. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: EUROCRYPT 2013 [42]. https://doi.org/10.1007/978-3-642-38348-9_9
77. Renauld, M., Standaert, F.X., Veyrat-Charvillon, N.: Algebraic side-channel attacks on the AES: Why time also matters in DPA. In: CHES 2009. pp. 97–111 (2009). https://doi.org/10.1007/978-3-642-04138-9_8
78. Rijmen, V., Daemen, J., Preneel, B., Bossalaers, A., De Win, E.: The cipher SHARK. In: FSE'96 (1996). https://doi.org/10.1007/3-540-60865-6_47
79. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: CHES 2010 (2010). https://doi.org/10.1007/978-3-642-15031-9_28
80. dos Santos, L.C., Gérard, F., Großschädl, J., Spignoli, L.: Rivain-prouff on steroids: Faster and stronger masking of the AES. In: Buhan and Schneider [21]. https://doi.org/10.1007/978-3-031-25319-5_7
81. Schneider, T., Moradi, A.: Leakage assessment methodology - extended version. JCEN **6**(2) (2016). https://doi.org/10.1007/s13389-016-0120-y
82. Standaert, F.: How (not) to use welch's t-test in side-channel security evaluations. In: CARDIS (2018). https://doi.org/10.1007/978-3-030-15462-2_5
83. Standaert, F.X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: CHES 2008 (2008). https://doi.org/10.1007/978-3-540-85053-3_26
84. Standaert, F.X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: EUROCRYPT 2009 (2009). https://doi.org/10.1007/978-3-642-01001-9_26
85. Standaert, F.X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The world is not enough: Another look on second-order DPA. In: ASIACRYPT 2010 (2010). https://doi.org/10.1007/978-3-642-17373-8_7
86. Veyrat-Charvillon, N., Gérard, B., Standaert, F.X.: Soft analytical side-channel attacks. In: ASIACRYPT 2014 (2014). https://doi.org/10.1007/978-3-662-45611-8_15
87. Whitnall, C., Oswald, E.: A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In: ASIACRYPT 2019 (2019). https://doi.org/10.1007/978-3-030-34618-8_9