

Generalized Feistel Ciphers for Efficient Prime Field Masking

Lorenzo Grassi¹, Loïc Masure², Pierrick Méaux³,
Thorben Moos⁴, François-Xavier Standaert⁴

¹ Ruhr University Bochum, Bochum, Germany

² LIRMM, Univ. Montpellier, CNRS

³ Luxembourg University, Esch-sur-Alzette, Luxembourg

⁴ Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium

Abstract A recent work from Eurocrypt 2023 suggests that prime-field masking has excellent potential to improve the efficiency vs. security tradeoff of masked implementations against side-channel attacks, especially in contexts where physical leakages show low noise. We pick up on the main open challenge that this seed result leads to, namely the design of an optimized prime cipher able to take advantage of this potential. Given the interest of tweakable block ciphers with cheap inverses in many leakage-resistant designs, we start by describing the **FPM** (Feistel for Prime Masking) family of tweakable block ciphers based on a generalized Feistel structure. We then propose a first instantiation of **FPM**, which we denote as **small-pSquare**. It builds on the recent observation that the square operation (which is non-linear in \mathbb{F}_p) can lead to masked gadgets that are more efficient than those for multiplication, and is tailored for efficient masked implementations in hardware. We analyze the mathematical security of the **FPM** family of ciphers and the **small-pSquare** instance, trying to isolate the parts of our study that can be re-used for other instances. We additionally evaluate the implementation features of **small-pSquare** by comparing the efficiency vs. security tradeoff of masked FPGA circuits against those of a state-of-the-art binary cipher, namely **SKINNY**, confirming significant gains in relevant contexts.

1 Introduction

The design of symmetric cryptographic algorithms is generally oriented towards optimizing their efficiency vs. security tradeoff. For most general applications, this has led researchers to focus primarily on binary ciphers with efficient bit-slice implementations, which are generally efficient in software [16] and hardware [57]. This trend has even been amplified when considering side-channel attacks, in good part due to the emergence of masking as the most popular solution to mitigate such attacks. While various types of masking schemes exist (e.g., additive [29], multiplicative [43], affine [42], polynomial [74], inner product [4], code-based [89]), the efficiency of Boolean masked implementations in software [44] and hardware [49] make it for now a default solution. As a result, ciphers optimized towards low AND complexity, enabling efficient bit-oriented

implementation (e.g., bitslicing), appeared for a while as the best approach [50]. This situation is also reflected by the recent NIST Lightweight Cryptography standardization effort, where most ciphers designed with leakage in mind (including the winner Ascon [35]) have efficient bitslice representations.¹

While it has been shown that Boolean masking can bring high security at limited cost, it is also known to suffer from practical limitations. Among others, it is only effective in contexts where leakages are sufficiently noisy [73,37,38], a condition that was shown to be challenging to reach without dedicated noise generation circuitry, both in software [5,24] and in hardware [72,69]. Building on theoretical advances of Dziembowski et al. [41], it has then been observed that computing in groups of prime order can significantly reduce the noise requirements of masking security proofs while keeping most of the benefits of additive encodings, and even providing security gains in the context of noisy leakages (that were not covered by theoretical analysis) [67]. More precisely, Masure et al. showed at Eurocrypt 2023 that for concretely-relevant leakage functions, prime-field masking can be quite efficient by re-using simple additions and multiplication algorithms “à la ISW” [54], and that the mild performance overheads due to operating in prime fields can be largely compensated by concrete side-channel security gains. Informally, these gains can be viewed as the result of a decreased “algebraic compatibility” between the leakage functions observed in practice (which are typically close to a linear combination of bits [76]) and the field in which we mask. For example, it is well-known that observing the least significant bit of Hamming weight leakages obtained from Boolean shares leads to information about the secret independent of the number of shares [80]. Moving to prime encodings, such an attack is not directly possible anymore because partial uncertainty “diffuses” better when combining the shares.

So far, this potential advantage of prime-field masking for counteracting side-channel attacks was only demonstrated for a toy AES-like cipher. The main open challenge that we pick up in this paper is, thus, the design of a dedicated lightweight cipher optimized for prime masking to enable fair comparisons with binary ciphers which are tailored for cost-efficiency when masked.

Given the interest of Tweakable Block Ciphers (TBCs) with cheap inverse for leakage-resistant modes of operation [7,10], we start by describing the FPM (Feistel for Prime Masking) family of tweakable block ciphers based on a generalized Feistel structure [71,52]. Among other advantages, TBCs allow reducing the need of idealized assumptions that are hard to justify in physical security analyzes and to minimize the side-channel attack surface during tag verification (which can leak in an unbounded manner thanks to the inverse trick of [11]). The FPM family of ciphers allows tweaks of variable size including a version without tweak (i.e., a block cipher, in order to enable comparisons with generic constructions [88]). It relies on a variant of the TWEAKEY framework [56], taking advantage of the fact that for most leakage-resistant modes of operation, the tweak is public information and requires no countermeasures (so we can actually use a simple key scheduling algorithm and a more complex, non-linear, tweak

¹ <https://csrc.nist.gov/Projects/lightweight-cryptography>

scheduling algorithm). While moving towards a first instantiation of FPM, we additionally exploit recent results from CHES 2023 which show how to obtain a secure implementation of the square operation (non-linear in \mathbb{F}_p) which is more efficient than a secure multiplication [27]. This provides natural incentive for designing a cipher using the square operation as only source of non-linearity, which further motivates the use of Feistel-like structures for FPM TBCs and their underlying building blocks, since the square is also non-invertible in \mathbb{F}_p . What then mostly remains is to choose the prime number defining the field in which we operate. Following [67], we use a Mersenne prime for efficiency reasons.² We set this modulus to $2^7 - 1$ in order to propose an instance tailored for secure hardware implementation, which we denote as **small-pSquare**.

Besides defining the FPM family of ciphers and a first instance, we provide an initial mathematical security analysis in order to select the number of cipher rounds of **small-pSquare**. Doing so, we try to separate the parts of the analysis that are generic (and could be re-used for other instances) from the ones that are linked to our choice of square S-box and 7-bit prime. Most importantly, we then compare masked FPGA implementations of **small-pSquare** and similar implementations of a binary cipher protected with Boolean masking. We use SKINNY for this purpose [6], which is a popular family of ciphers with tweakable versions that amongst other applications was used in Romulus, a finalist to NIST lightweight cryptography competition, and for which a rich literature on the construction and analysis of state-of-the-art Boolean masked implementations exist, both automated [58] and hand-made [85]. Our experiments allow us to confirm the excellent performances and significantly improved efficiency vs. (side-channel) security tradeoff for **small-pSquare**. We show in Section 6 that while unprotected **small-pSquare** implementations come with overheads (compared to SKINNY), these overheads vanish in the context of masked implementations where both algorithms perform similarly. As expected, **small-pSquare** also has significantly improved performances compared to the toy AES-like cipher considered in [67]. Furthermore, we show in Section 7 that for similar architectures, **small-pSquare** offers side-channel security levels which exceed those of masked SKINNY implementations by (at least) one order of magnitude.

We conclude the paper by discussing scopes for further research and other instances of FPM ciphers. First, considering different implementation contexts, for example **mid-pSquare** variants could be relevant to investigate for FPGAs with DSP blocks (e.g., with a 17-bit prime) or for ARM Cortex-like devices (e.g., with a 31-bit prime). Second, and more prospectively, **big-pSquare** variants (with larger primes) could be of interest conceptually due to their similarity with the different prime ciphers developed for other applications (e.g., fully-homomorphic encryption, multi-party computation, zero-knowledge proofs), in order to better understand the differences and similarities between the design goals to optimize for these various applications [2,1,3,46,36,30], and possibly to offer stronger physical security guarantees thanks to the larger field computations [40].

² Both because such prime numbers allow very efficient modular reductions and because the $x \mapsto 2x$ operation is a rotation of the bits that is free in hardware.

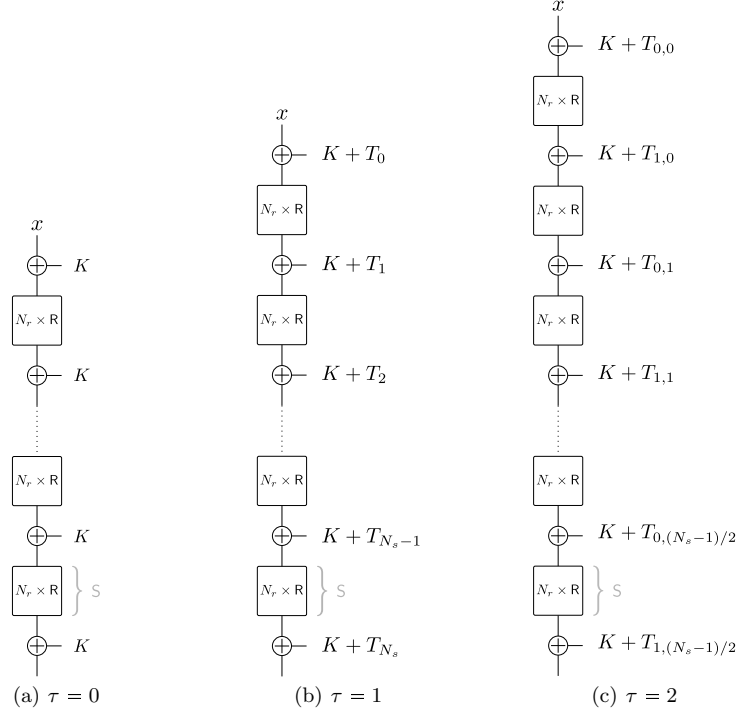


Figure 1: High-level view of FPM (tweakable) ciphers. We use the shortcut notation $N_r \times R$ to denote the application of the round R N_r times.

2 Feistel for Prime Masking

In this section, we introduce the FPM_τ family of TBCs. We start by describing the high-level Feistel structure we use in Section 2.1. We then detail the internal components of this structure in Sections 2.2 and 2.3. We conclude by summarizing the design space that this family of ciphers defines in Section 2.4.

2.1 High-Level Structure

Let $p \geq 3$ be a prime number, and let $n = 2 \cdot n' \geq 4$ be an integer. The high-level structure of FPM_τ TBCs is given in Figure 1. FPM_τ ciphers take as inputs a plaintext $x \in \mathbb{F}_p^n$, a key $K \in \mathbb{F}_p^n$ and an optional tweak defined as:

$$T := \begin{cases} (T^{(1)}, T^{(2)}, \dots, T^{(\tau)}) \in \mathbb{F}_p^{\tau n} & \text{if } \tau \geq 1 \\ \emptyset & \text{otherwise } (\tau = 0) \end{cases}.$$

If $\tau = 0$, then FPM_0 ciphers receive no tweak input and correspond to block ciphers. FPM_τ ciphers are key alternating ciphers, where a tweakkey is added every $r > 1$ rounds. We denote a single round as R , and we denote a group of N_r rounds R as a step S . A tweakkey addition is performed after every step. If $\tau = 0$, the tweakkey is always the master key K . If $\tau \geq 1$, the tweakkeys are

defined as $K + T_{0,0}, K + T_{1,0}, \dots, K + T_{\tau-1,0}, K + T_{0,1}, K + T_{1,1}, \dots, K + T_{\tau-1,1}, \dots, K + T_{0,i}, K + T_{1,i}, \dots, K + T_{\tau-1,i}$, where the values $T_{j,i} \in \mathbb{F}_p^n$ are produced by a tweak scheduling and are independent of the master key. If $\tau = 1$, we usually omit the first index for simplicity (i.e., we use T_i instead of $T_{0,i}$).

The rounds $R : \mathbb{F}_p^n \rightarrow \mathbb{F}_p^n$ (and the steps S) are independent of both the tweak and of the master key. The number of rounds per step N_r must at least guarantee that full diffusion is achieved in the steps. Regarding the number of steps N_s , we additionally require that if $\tau \geq 1$, then $N_s - 1$ must be a multiple of τ in order to guarantee that the tweak is absorbed in equal measure.

Key and tweak scheduling algorithms. Since we do not claim security against related-key attacks, we opted for the simplest key scheduling algorithm, which consists of having all the subkeys equal to the master key. Note that several tweakable lightweight symmetric primitives in the literature are based on similar design choices, including SKINNY [6] (which we will use in our comparisons).

By contrast, our designs make use of a tweak scheduling algorithm. As mentioned in the introduction, this is because in many leakage-resistant modes of operation, the tweak is public and therefore does not require any protection against leakage. This context calls for operations that are cheap to implement without countermeasures (in hardware and software) while providing good cryptographic properties. Since FPM ciphers operate in prime fields, a natural candidate for this purpose is to combine a shuffling of the \mathbb{F}_p -words in each state with an invertible mapping of the “bits” in each \mathbb{F}_p -word, for example taking advantage of the fact that linear mappings in \mathbb{F}_2 are non-linear in \mathbb{F}_p . More precisely, for each $i \geq 0$ and for each $j \in \{0, 1, \dots, \tau - 1\}$, we define $T_{j,i}$ as:

$$T_{j,i} := \Psi_0 \left(T_{\Pi^i(0)}^{(j)} \right) \parallel \Psi_1 \left(T_{\Pi^i(1)}^{(j)} \right) \parallel \dots \parallel \Psi_{n-1} \left(T_{\Pi^i(n-1)}^{(j)} \right) \in \mathbb{F}_p^n,$$

where:

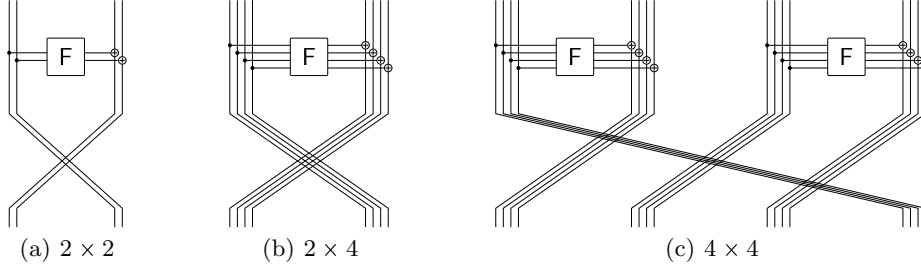
- For each $l \in \{0, 1, \dots, n-1\}$, $T_l^{(j)} \in \mathbb{F}_p$ denotes the l -th \mathbb{F}_p -word of $T^{(j)}$;
- Π is a shuffling of $\{0, 1, \dots, n-1\} \subseteq \mathbb{N}$ satisfying the following conditions:
 1. Π^i (where Π^i denotes the application of Π i consecutive times) is different from the identity for each $i \leq i'$ and a sufficiently large i' ;
 2. Π does not contain fix points and (if possible) two consecutive elements before the shuffling are not consecutive after it.
- For each $l \in \{0, 1, \dots, n-1\}$, $\Psi_l : \mathbb{F}_p \rightarrow \mathbb{F}_p$ is an invertible mapping.

As we are going to show in the next section, this tweak scheduling algorithm allows us to adapt the simple security arguments used in [51] to our scheme.

2.2 Rounds R of FPM_τ via Type-II Generalized Feistel

Let $b, c \geq 2$ be positive integers such that $n = b \cdot c$ and $b = 2 \cdot b'$. The rounds R over $(\mathbb{F}_p^c)^b \equiv \mathbb{F}_p^n$ of FPM_τ ciphers are based on a Type-II generalized Feistel network structure [90,71,52]. They are defined as:

$$(x_0, x_1, \dots, x_{b-2}, x_{b-1}) \mapsto (F(x_0) + x_1, x_2, F(x_2) + x_3, \dots, F(x_{b-2}) + x_{b-1}, x_0),$$

Figure 2: FPM rounds. Left: $b=2, c=2$; middle: $b=2, c=4$; right: $b=4, c=4$.

where $F : \mathbb{F}_p^c \equiv \mathbb{F}_{p^c} \rightarrow \mathbb{F}_p^c \equiv \mathbb{F}_{p^c}$ is discussed in the next subsection. Such structures are characterized by two parameters: $b = 2 \cdot b' \geq 2$ is the number of branches in the generalized Feistel network (where each branch can carry several values in \mathbb{F}_p), $c \geq 2$ is the number of values in \mathbb{F}_p of each branch, that the non-linear F function takes as input. Figure 2 illustrates three examples.

Remark: Achieving faster diffusion. In this paper, we limit ourselves to instances with small b values (up to 4) and to the classical Type-II generalized Feistel scheme, in which a shift is applied at the output of the non-linear layer. However, several studies have been conducted in literature to find better shuffles of the words that can achieve faster diffusion for larger b values (see for example [83,28]), which we suggest to use in place of the shift whenever applicable. Besides the minimum number of rounds necessary for achieving full diffusion, these references also provide the number of active functions F .

2.3 Function F of the Type-III Generalized Feistel

The F functions over \mathbb{F}_p^c are designed to 1) be bijective (since collisions at their outputs could make the security analysis harder) and 2) ensure full non-linear diffusion. This is achieved by combining the following components:

- A first non-linear layer is instantiated via a Type-III generalized Feistel network [90,71,52] (without the shift) of the form: $(x_0, x_1, x_2, \dots, x_{c-1}) \in \mathbb{F}_p^c \mapsto (x_0, G_{0,0}(x_0) + x_1, G_{1,0}(x_1) + x_2, \dots, G_{c-2,0}(x_{c-2}) + x_{c-1}) \in \mathbb{F}_p^c$, where $G_{0,0}, G_{1,0}, \dots, G_{c-2,0}$ are non-linear operations over \mathbb{F}_p .
- The non-linear layer is followed by a multiplication with a $c \times c$ Maximum Distance Separable (MDS) matrix [33,34], typically lightweight [66,60,39].³
- Finally, a non-linear layer instantiated via a Type-III generalized Feistel network (with the shift) is applied to the state: $(x_0, x_1, x_2, \dots, x_{c-1}) \in \mathbb{F}_p^c \mapsto (x_0, G_{0,1}(x_0) + x_1, G_{1,1}(x_1) + x_2, \dots, G_{c-2,1}(x_{c-2}) + x_{c-1}) \in \mathbb{F}_p^c$, where $G_{0,0}, G_{1,0}, \dots, G_{c-2,0}$ are again non-linear operations over \mathbb{F}_p .

Before the application of each Type-III generalized Feistel network, a round constant is added on the first element x_0 . We suggest to generate these constants

³ The branch number \mathcal{B} of a matrix over \mathbb{F}_p^t is defined as $\mathcal{B}(M) = \min_{x \in \mathbb{F}_p^t \setminus \{0\}} \{ \text{hw}(x) + \text{hw}(M(x)) \}$, where $\text{hw}(\cdot)$ is denoted as the bundle weight in wide trail terminology. A matrix $M \in \mathbb{F}_p^{t \times t}$ is an MDS matrix if and only if $\mathcal{B}(M) = t + 1$.

via bit rotations of a mathematical constant as $\pi = 3.14159 \dots \in \mathbb{R}$ rounded to a bit size that is large enough to avoid cycles for the number of cipher rounds.

We note that using a Type-III generalized Feistel network for the F functions is motivated by the fact that a potential candidate for the $G_{i,j}$ functions is the square operation which is non-invertible (or other small power maps which are non-bijective in the respective field). In case the $G_{i,j}$ functions are themselves bijective, a simpler alternative is to directly use SPN rounds.

We also note that since we use a bijective F , exploiting a Feistel structure for the rounds of Section 2.2 is not mandatory (e.g., an SPN could work there too). However, it has the advantage that the F function can be chosen without any regard for the implementation efficiency of its inverse (with or without masking), which would not be the case when used as an S-box in a typical SPN construction. Furthermore, the Feistel strategy directly enables us to obtain cheap inverses in the sense that 1) cost of decryption \approx cost of encryption and 2) implementing a hardware circuit that can both encrypt and decrypt is not (significantly) more expensive than one which can only encrypt (in contrast to most standard SPN designs). In general, we believe that the high-level structure of FPM ciphers is a natural starting point given our goals. SPN-based structures would also require an additional linear layer (which may be more expensive) and it is unclear whether it would enable a reduction of the number of rounds by half (to compensate for the cost of operating on the full state in each round). Yet, investigating whether such prime SPN ciphers could potentially improve over the proposed FPM designs remains an interesting open problem.

2.4 Summary of the FPM_τ Design Space

The size of an FPM_τ cipher is determined by the number of tweaks τ , the prime integer p , the number of branches of the Type-II generalized Feistel network b and the number of input words of the F functions c . We use the notation $FPM_\tau(\rho, b, c)$ for this purpose, where $\rho = \lceil \log_2(p) \rceil$. The cipher specifications additionally require to choose the functions $G_{i,j} : \mathbb{F}_p \rightarrow \mathbb{F}_p$ and an MDS matrix, and to define the shuffling/mapping of the tweak scheduling algorithm and the round constants.⁴ Next, we first provide high-level security arguments that justify the design choices of FPM ciphers in Section 3. We then propose a first hardware-oriented instance in Section 4 for which we analyze the mathematical security in Section 5 and the implementation efficiency & security in Sections 6 and 7.

3 High-level Rationale and Security Arguments

We now provide a high-level rationale and security arguments for FPM_τ TBCs.

3.1 TWEAKEY Framework and LED-like Design

FPM_τ ciphers follow the TWEAKEY framework proposed by Jean et al. [56] at Asiacrypt'14. In contrast to the majority of the TBCs following this framework

⁴ Variants where the F function uses nearly MDS matrices could be considered.

(including SKINNY), we add the tweak only every $N_r > 1$ rounds, where N_r is strictly bigger than 1. This approach is not new in the literature, as it has been already exploited in the block cipher LED [51]. Its main advantage is to allow a very simple security analysis concerning related-tweak attacks.

More precisely, since the tweaks are public, the attacker can always control them. Similar to a related-key attack [15], in a related-tweak attack the attacker encrypts (resp., decrypts) the same or different plaintext(s) (resp., ciphertexts) under several related tweaks. (Anticipating the detailed analysis of Section 5, we emphasize that related-tweak attacks are usually based on statistical properties and not on algebraic ones.) A possible way to avoid such attacks is to treat the tweaks exactly as the plaintexts. That is, not to make any distinction between plaintexts and tweaks. This is what is done in a sponge/duplex construction [12,13], but it requires a larger state in order to arrange the inner part, which is not suitable in our case. Another approach to prevent related-tweak attacks is the one proposed in [51] to prevent related-key attacks. That is, adding the tweak every $N_r > 1$ rounds. The argument for $\tau = 1$ is relatively simple:

- A statistical attack as the differential one [18,19] exploits the probability distribution of a non-zero input difference leading to an output difference after a given number of rounds. The security is achieved if the probability of any differential characteristic is much smaller than the security level;
- Given $T \in \mathbb{F}_p^n$, assume for simplicity that all the T_i 's $\in \mathbb{F}_p^n$ are equal to T ;
- If a difference is inserted in the tweak, then every sub-tweak T_i will be active;
- Hence, it is impossible to force two consecutive steps S to be non-active (i.e., with zero input and zero output differences). That is, for every two consecutive steps S of N_r -rounds, at least one of them must be active.

Indeed, let's assume that the output difference of the i -th step S coincide with the difference in T_i . In this case, the next $i + 1$ -th step S is not active, since its input difference is equal to zero. But the next tweak T_{i+1} will introduce again the difference, making the next $i + 2$ -th step S active. Using the number of active steps N_s (each one composed of N_r rounds), it is therefore possible to provide simple security arguments for preventing differential and other statistical attacks, which reduce to the security of the public permutation S (which is independent of the tweaks and the master key). We refer to the full version of this paper [47] for an initial analysis (based on published results) regarding the selection of the number of steps N_s independently of their internal structure.

The previous argument can be generalized for a non-trivial tweak scheduling $T : \mathbb{F}_p^{\tau \cdot n} \equiv (\mathbb{F}_p^n)^\tau \rightarrow (\mathbb{F}_p^n)^*$, for example if the following properties are satisfied: 1) T is bijective, and 2) $T_{j,\cdot} \in \mathbb{F}_p^n$ is active if and only if $T^{(j)} \in \mathbb{F}_p^n$ is active. Equivalently, this second condition is satisfied if there exist τ invertible maps $T_0, T_1, \dots, T_{\tau-1}$ over \mathbb{F}_p^n such that $T_{j,i} = T_j^{i+1}(T^{(j)})$ for each $i \geq 0$, where $T_j^{i+1} := T_j \circ T_j \circ \dots \circ T_j$ for i times. (We emphasize that this is not a necessary condition.) In our case, we achieve this property by defining $T_{j,\cdot}$ via a shuffle of the \mathbb{F}_p -words of $T^{(j)}$. (The mapping of each \mathbb{F}_p does not affect this property.) A detailed argument will be given for small-pSquare with $\tau = 1, 2$ in Section 5.1.

We leave the question whether adding the tweak every round could lead to improved (but harder to analyze) security as a scope for further research.

3.2 Rationale behind the Generalized Type-II Feistel Scheme

The main motivation behind the choice of defining FPM_τ ciphers based on a generalized Feistel structure relates to the goal of having TBCs with cheap inverses that are useful in some leakage-resistant modes of operation [7,10]. This result can be achieved via 1) a Feistel or Lai-Massey scheme [62,84,45], 2) an SPN scheme with the “reflection” property like *Prince* [22,23,14], or 3) an SPN scheme in which every round – without the constant additions – is an involution (that is, $R = R^{-1}$) like *Noekon* [32], *Khazad* [82] or *Iceberg* [82]. Even if all options are valid from a security viewpoint, the first one comes with the least constraints on its internal components, which is desirable in our setting in order to enable these components to be selected primarily for their properties against leakage. After discarding Type-I Feistel schemes that require too many rounds for achieving full diffusion, we opted for Type-II generalized Feistel networks instead of Type-III ones. As witnessed by designs like *Hight* [53] or *Clelia* [79], they generally offer a good security vs. efficiency compromise.

3.3 Rationale and Construction of the Function F

As mentioned in Section 2.3, the F functions aim to ensure good non-linear diffusion while remaining bijective (in order to simplify the security analysis). For functions $G_{i,j}$ over \mathbb{F}_p that are themselves bijective, this could be directly obtained with two SPN rounds. Yet, and as mentioned in the introduction, one natural candidate $G_{i,j}$ function is the square power map, which leads to efficient masked implementations [27]. As a result, we opted for F functions based on two rounds of a generalized Feistel network.⁵ We selected the Type-III version which is more similar to SPNs in terms of their number of non-linear $G_{i,j}$ functions and replaced the middle shift of the \mathbb{F}_p -words by an invertible linear layer in order to speed up the non-linear diffusion, an idea that resembles the one in [8].

Regarding the choice of the linear layer, we opted for an MDS matrix which allows to achieve full non-linear diffusion over \mathbb{F}_p^c in only two rounds. Examples include lightweight candidates [66,60,39] adapted to the prime case (where the multiplication per two can be cheap – see Footnote 2). Such MDS matrices could be replaced by any invertible matrix with a smaller branch number that allows to get full non-linear diffusion in two rounds, as the ones in [65].

Finally, the round constant additions aim to (i) differentiate the rounds (e.g., for preventing slide attacks [20,21]), (ii) break any fixed points, and (iii) break any invariant subspace [63,64,48]. Since $x \mapsto x^2$ has only two fixed points (namely, 0 and 1) and since \mathbb{F}_p does not have any non-trivial subspace (as opposed to $\mathbb{F}_{2^t} \equiv \mathbb{F}_2^t$), we believe that one \mathbb{F}_p -constant addition before each non-linear Type-III generalized Feistel layer is sufficient. As an extra condition when using Mersenne primes, we require that the round constants do not belong to

⁵ For instances relying on invertible $G_{i,j}$ functions, we suggest using two SPN rounds.

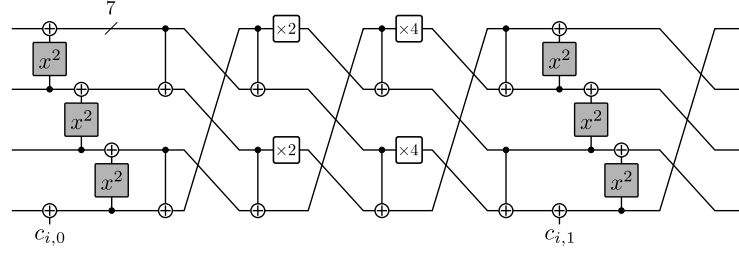


Figure 3: F-function used in small-pSquare.

any subspace of \mathbb{F}_2^p (where $p = 2^\rho - 1$). The choice to generate them via a bit rotation of a fixed mathematical constant like π is for efficient (hardware) implementation purposes. The mathematical constant must be chosen such that all the rotations are in $\{0, 1, \dots, p-1\}$ where p is the prime that defines \mathbb{F}_p .

4 small-pSquare: a Hardware-oriented Instance

In this section, we provide the specifications of a first instance of an FPM_τ cipher. As mentioned in the introduction, its high-level rationale follows two main guidelines. First, we aim to exploit the recently proposed secure squaring gadgets from [27], which were shown to be more efficient than secure multiplications in \mathbb{F}_p . As a result, we use the square as power map for the $G_{i,j}$ functions of Section 2.3. Second, we aim to enable efficient hardware implementations. As a result, we use a small Mersenne prime $p = 2^7 - 1$. We then propose to use the rounds depicted in the right part of Figure 2, leading to a $\text{FPM}_\tau(7, 4, 4)$ cipher that provides $\approx 7 \times 4 \times 4 = 112$ bits of security and we denote as **small-pSquare**.

We first detail the different components of the function F (depicted in Figure 5), then finalize the specification of the tweak scheduling algorithm and conclude with the suggested number of rounds per steps and steps.

Non-Linear Layer. The non-linear layer of **small-pSquare** is instantiated with the following $\mathbb{F}_{2^7-1}^4 \rightarrow \mathbb{F}_{2^7-1}^4$ mapping:

$$(x_0, x_1, x_2, x_3) \mapsto (x_0 + x_1^2, x_1 + x_2^2, x_2 + (x_3 + c_{i,j})^2, x_3 + c_{i,j}) \quad ,$$

where $c_{i,j}$ is a round constant specified thereafter.

Linear Layer. The linear layer of **small-pSquare** is instantiated with the invertible matrix $M \in \mathbb{F}_{2^7-1}^{4 \times 4}$ defined as:

$$M = \begin{bmatrix} 3 & 2 & 1 & 1 \\ 7 & 6 & 5 & 1 \\ 1 & 1 & 3 & 2 \\ 5 & 1 & 7 & 6 \end{bmatrix} \quad .$$

This matrix has been introduced by Duval et al. [39] and is MDS over \mathbb{F}_{2^7-1} . It can be implemented as a Type-II Feistel-like construction as shown in Figure 5, with only 8 additions and a depth of 4 (which is optimal for 8 additions). We

recall that the doubling operation (i.e., $x \mapsto 2 \cdot x$) modulo a Mersenne prime is just a bit rotation, hence free in hardware and cheap in software.

Round Constants. The first 64 bits of the binary sequence of π are (in hexadecimal): $\pi_{\text{bin64}} = 0\text{x}C90FDAA22168C234$. Let us denote the bit-wise rotation left via \ll . Then, the left F-function at round i uses the round constants:

- $c_{i,0} = (\pi_{\text{bin64}} \ll i) \bmod 2^7$,
- $c_{i,1} = (\pi_{\text{bin64}} \ll (i + 16)) \bmod 2^7$,

while the right F-function at round i uses the round constants:

- $c_{i,2} = (\pi_{\text{bin64}} \ll (i + 32)) \bmod 2^7$,
- $c_{i,3} = (\pi_{\text{bin64}} \ll (i + 48)) \bmod 2^7$.

As no sequence of 7 consecutive 1s exists in π_{bin64} , all $c_{i,0}, c_{i,1}, c_{i,2}, c_{i,3}$'s $\in \mathbb{F}_{2^7-1}$.

Tweak Scheduling ($\tau \geq 1$ Only). Let Π_{16} be the shuffle of the 16 \mathbb{F}_{2^7-1} -words in the tweak schedule sub-tweak word permutation defined as:

$$\Pi_{16}(x_0 \| x_1 \| \dots \| x_{15}) = x_9 \| x_5 \| x_{13} \| x_{15} \| x_{12} \| x_7 \| x_{14} \| x_2 \| x_4 \| x_6 \| x_8 \| x_3 \| x_{10} \| x_1 \| x_{11} \| x_0.$$

Π_{16} has a cycle period of 140 which is the largest we found for 16-element shuffles (and more than sufficient for our envisioned step numbers). On each \mathbb{F}_{2^7-1} -word, we then apply a bit-shuffle Ψ_l defined as

$$\forall l \in \{0, 1, \dots, 15\} : \quad \Psi_l(x) = \psi_7(2^l \cdot x \bmod 2^7),$$

where the multiplication with 2^l corresponds to a shift of the bits of l positions (when working over \mathbb{F}_2^7), and where ψ_7 is defined as:

$$\psi_7 \left(x = \sum_{i=0}^6 x_i \cdot 2^i \right) = x_0 \cdot 2^5 + x_1 \cdot 2^3 + x_2 \cdot 2^0 + x_3 \cdot 2^4 + x_4 \cdot 2^1 + x_5 \cdot 2^6 + x_6 \cdot 2^2,$$

for each $x \in \mathbb{F}_{2^7-1}$ where $x_0, x_1, \dots, x_6 \in \{0, 1\}$. The cycle period of ψ_7 is 12 (i.e., the maximum possible for a permutation over 7 bits). Moreover, the polynomial corresponding to ψ_7 over \mathbb{F}_{2^7-1} is of degree 125 (i.e., the maximum possible) and contains 46 out of the 127 monomials possible. We refer to the full version of this paper [47] for details on ψ_7 , where we also prove that the polynomial over $\mathbb{F}_{2^{\rho}-1}$ corresponding to any bit shuffling only contains monomials of odd degree.

Number of rounds. For a security level of 112 bits and the aforementioned parameters ($p = 2^7 - 1; b = 4, c = 4$), we use $N_r = 4$ rounds per step and we use $N_s = 9$ steps for $\tau = 0$, $N_s = 16$ steps for $\tau = 1$ and $N_s = 21$ steps for $\tau = 2$. The security analysis that supports these choices is given in the next section.

5 Mathematical Security Analysis of **small-pSquare**

We now evaluate the security of **small-pSquare** against standard attack vectors, including classical and truncated differential attacks and algebraic attacks (based on interpolation, linearization, higher-order differentials and Gröbner bases). We describe the attacks having a larger impact on **small-pSquare**'s number of rounds in the paper. Details of further attacks are presented in the full version [47].

Overview of the attacks. As we are going to show, the main attack vector against **small-pSquare** is differential cryptanalysis [18,19], which we present in detail in Section 5.1 for the case $\tau = 0$ and the case $\tau \geq 1$ (for which we consider related-tweak differential attacks). In this last case, we exploit the strategy introduced by the LED designers and recalled in Section 3.1 for guaranteeing security against related tweaks. Truncated [59] and impossible differential [17] cryptanalysis as well as other statistical attacks including linear cryptanalysis [68] and boomerang attacks [87] are detailed in the full version [47]. Contrary to MPC-/FHE-/ZK-friendly schemes defined over prime fields, and similar to classical/traditional symmetric primitives, algebraic attacks are not the main threat against **small-pSquare** in our analysis, essentially due to the small size of the prime $p = 2^7 - 1$ and the high number of variables $n = 16$. For this reason, we discuss the degree and density of the polynomial representation of **small-pSquare** in Section 5.2, limit ourselves to linearization attacks [31] in Section 5.3, while we defer the description of interpolation [55], higher-order differential [61,59] and Gröbner bases based attacks to the full version of the paper [47].

We mention that in all these cases, we tried to identify concrete strategies on how to speed up the attacks by making use of related tweaks. In particular, we propose concrete ways to use related tweaks for speeding up algebraic attacks which, to the best of our knowledge, has not been thoroughly studied yet in the open literature.⁶ Besides, the non-linear tweak scheduling algorithm of **small-pSquare** is also aimed to frustrate such improved cryptanalysis attempts.

5.1 Differential Cryptanalysis

Given pairs of inputs with some fixed input differences, differential cryptanalysis [18,19] considers the probability distribution of the corresponding output differences produced by the cryptographic primitive. Let $\Delta_I, \Delta_O \in \mathbb{F}_p^n$ be respectively the input and the output differences through a permutation P over \mathbb{F}_p^n . The Differential Probability (DP) of having a certain output difference Δ_O given a particular input difference Δ_I is equal to:

$$\text{Prob}_P(\Delta_I \rightarrow \Delta_O) = \frac{|\{x \in \mathbb{F}_p^n \mid P(x + \Delta_I) - P(x) = \Delta_O\}|}{p^n}.$$

⁶ Binary schemes like AES or SHA-3/Keccak have been shown vulnerable to statistical attacks mainly, while algebraic attacks gain more popularity recently due to the raising of symmetric schemes designed for applications as MPC, FHE, and ZK. Still, symmetric schemes designed for such applications are not tweakable designs.

In the case of iterated schemes, a cryptanalyst searches for ordered sequences of differences over any number of rounds that are called differential characteristics/trails. Assuming the independence of the rounds, the DP of a differential trail is the product of the DPs of its one-round differences.

Differential property of F. As first step, we compute the maximum differential probability of F. Since $x \mapsto x^2$ is a quadratic map, $\text{DP}_{\max}(x \mapsto x^2) = p^{-1} \approx 2^{-7}$. For our goal, we just need to compute the minimum number of active square maps in F. We can check that it corresponds to 2. Indeed, let $(x_0, x_1, x_2, x_3) \mapsto (x_0, x_0^2 + x_1, x_1^2 + x_2, x_2^2 + x_3)$ be the Feistel layer, and keep in mind that the branch of the MDS matrix M is 5. Then:

- If no square map is active in the first Feistel layer of F (hence, x_3 is the only active component), then only one output is active at its output. After the multiplication with the MDS matrix, all inputs of the second Feistel layer are active, which implies the result. A similar result holds if only one square map is active in the first Feistel layer, which corresponds to the case in which only one among $\{x_0, x_1, x_2\}$ is active. In such a case, two outputs of the first Feistel layer are active, which implies that at least 3 inputs of the second Feistel layer are active and at least 3 square maps are active for each F;
- If two inputs are active, the best scenario for the attacker occurs when the active inputs are either $\{x_0, x_3\}$ or $\{x_1, x_3\}$. In this case, exactly one square map is active in the first Feistel layer (due to the fact that x_3 does not activate any square map), and three outputs of the Feistel layer are active (due to the fact that x_0, x_1 and x_3 are not consecutive). Since the matrix is MDS, then at least two inputs are active for the second Feistel layer, which implies that at least one square map is active in the second Feistel layer. As a result, at least 2 square maps are active for each F;
- If 3 or 4 inputs are active, at least two square maps are active for each F.

Overall, it directly follows that $\text{DP}_{\max}(x \mapsto F(x)) = p^{-2} \approx 2^{-14}$.

Differential property of S. As shown in [83,28], at least four consecutive rounds of Type-II generalized Feistel network are necessary to reach full diffusion (i.e., $N_r \geq 4$). Over 4 consecutive rounds, at least 3 functions F are active (as visually illustrated in the full version [47]). As a result, by setting $N_r = 4$, it directly follows that $\text{DP}_{\max}(x \mapsto S(x)) \leq \text{DP}_{\max}(x \mapsto S(x))^3 \leq p^{-6} \approx 2^{-42}$.

Number of steps for security. Finally, we compute the minimum number of steps N_s for guaranteeing security. Due to clustering effect (that is, due to the fact that several differential characteristics can be used together for setting up the attack) and due to the possibility to exploit a Meet-in-the-Middle approach for setting up the attack, we claim that the scheme is secure if every differential characteristic has probability smaller than $2^{-2.5 \cdot \kappa} \approx 2^{-280}$ for an arbitrary factor 2.5 ,⁷ where $\kappa = 112 = 7 \cdot 16$ is our target security level. Moreover, we conjecture that the attacker cannot skip more than 2 steps S by a simple partial key-guessing, since one step S is sufficient for achieving full diffusion.

⁷ We take inspiration on the AES-128, which has $10 = 2.5 \cdot 4$ rounds, where 4 is the minimum number of rounds for preventing classical differential attacks.

Case: $\tau = 0$. By simple computation, we have $N_s \geq \lceil 280/42 \rceil + 2 = 7 + 2 = 9$ where 2 steps S are added for preventing partial key-guessing strategies.

Case: $\tau = 1$. Following the argument proposed by LED’s designers in [51, Sect. 3] and recalled in Sect. 3.1, the attacker can choose related tweaks such that only one out of two consecutive steps S is active. As a result, it is sufficient to double the number of steps S obtained for $\tau = 0$ to guarantee security. That is, $N_s \geq 2 \cdot 7 + 2 = 16$, where we again add 2 steps S for preventing partial key-guessing.

Case: $\tau = 2$. In this case, the attacker has more freedom in the choice of the related tweaks. Still, we can adapt the previous security argument as follows. Let us consider separately the next two cases: (i) both $T^{(0)}$ and $T^{(1)}$ are active (hence, $T_{0,i}$ and $T_{1,i}$ are both active for each $i \geq 0$ due to the definition of the tweak schedule), and (ii) only one among $T^{(0)}$ and $T^{(1)}$ is active (hence, only one among $T_{0,i}$ and $T_{1,i}$ is active for each $i \geq 0$ due to the definition of the tweak schedule). In the first case, the analysis proposed for $\tau = 1$ applies, which implies that at least one among two consecutive steps S is active. In the second case, w.l.o.g., we assume that $T^{(0)}$ is active and $T^{(1)}$ is inactive. We introduce a “super-step” $S^2 := S \circ S$ as the application of two consecutive steps S . By working as before, we can deduce at least one among two consecutive super-steps S^2 is active. Moreover, if a super-step is active, then the two steps S that compose it are active. Indeed, the fact that S^2 is active implies that $T_{0,i}$ introduces the difference in the first S . Its output difference cannot be canceled by $T_{1,i}$, which is inactive due to the tweak schedule and due to the assumption. Hence, both steps of S^2 are active. The same result applies if $T^{(1)}$ is active and $T^{(0)}$ is inactive.

This reasoning implies that $N_s \geq 14$ is a necessary condition for security. Yet, we have to keep in mind that the attacker can potentially skip one super-step S^2 at each side of the cipher by working with input (respectively, output) differences in the plaintexts (resp., ciphertexts) that cancel out with the ones in the tweaks, leading S^2 to be inactive. As a result, we require that $N_s \geq 2 \cdot 7 + 4 + 2 = 20$, where we again add 2 steps S for preventing partial key-guessing.

5.2 Degree and Density of the Polynomial Representation

In general, algebraic attacks try to take advantage of the “simple” algebraic description of a scheme for breaking it, where the simplicity can relate (among other properties) to the low degree of the encryption/decryption function, the sparsity of the polynomial representation of such functions, or a particular structure of the algebraic system generated by the cipher. The main ingredient for preventing these attacks is the minimal number of rounds such that the polynomial representations of the cipher have a sufficient degree and too many monomials for the attacks to apply with a complexity lower than 2^κ . In this section, we therefore study these two characteristics, pointing out that the encryption function of **small-pSquare** with a fixed key and tweak could be analyzed as a mapping over $\mathbb{F}_{p^{16}}$ for $p = 2^7 - 1$. Nevertheless, since all the operations of \mathbf{F} are at the basis field level (squaring in \mathbb{F}_p), the field we consider for the cryptanalyses is \mathbb{F}_p , and the polynomials built by an adversary belong to $\mathbb{F}_p[x_0, \dots, x_{15}]/(x_0^p - 1, \dots, x_{15}^p - 1)$.

Note that similar results hold in the context of related tweaks, for which the adversary can consider the same polynomials but with more variables.

Growth of the degree. We first focus on the minimal degree that a polynomial in this representation can have, and the number of different monomials that appear in an algebraic system obtainable, after r rounds. Note that the degree in one variable is at most $p - 1$, the total degree is then at most $16(p - 1)$, and the total number of monomials is p^{16} . The degree of F is $\deg(F) = 4$. More precisely, it is 4 in three components and 2 in the last one. The degree of its inverse is $\deg(F^{-1}) = 8^2 = 2^6$. Indeed, note that the inverse of the internal function given by $(y_0, y_1, y_2, y_3) = (x_0 + x_1^2, x_1 + x_2^2, x_2 + x_3^2, x_3)$ is given by $(x_0, x_1, x_2, x_3) = (y_0 - (y_1 - (y_2 - y_3^2)^2)^2, y_1 - (y_2 - y_3^2)^2, y_2 - y_3^2, y_3)$.

In both cases (F and F^{-1}), we emphasize that one component of the internal function of F and F^{-1} has degree one only. Moreover, in the second case, we emphasize that the degree is different for each output variable, and that only one of them has actually maximum degree 8, and therefore 64 for F^{-1} . It follows that the degree of a step of r consecutive rounds S is $\deg(S) = \deg(F)^r = 2^{2 \cdot r}$, where we point out that half of the components have degree 1 at the end of the first round due to the Feistel structure. Accordingly, at round r two blocks have degree $\deg(F^{r-1}) = 2^{2 \cdot r - 2}$. Therefore the minimal degree a polynomial can have after r rounds is $2^{2 \cdot r - 3}$ until it reaches the maximum degree. Since the degree of the inverse of a Type-II Feistel scheme is equal to the degree of the Type-II Feistel scheme itself, the same bound applies for r consecutive steps S^{-1} .

Density of the polynomial representation. While the degree's growth is an important indicator in order to prevent algebraic attacks, another factor that plays a crucial role is the density of the polynomial representation. Indeed, various algebraic attacks depend on the number of monomials that appear in the polynomials, which implies that a scheme that admits a sparse polynomial is in general not secure against algebraic attacks even if it is of high degree.

Experimentally, we can verify the number of monomials we obtain in each of the 16 polynomials of a round, but even with simpler versions with a prime smaller than 127 it becomes too complex in practice after a few rounds. For example, even for $p = 3$, we observed by practical tests that we already get more than 2^{16} different monomials in some of the polynomials at the end of the third round. Since these experiments are quickly getting impractical, we decided instead to determine the number of rounds for which we expect each polynomial to be dense by considering the following approach. First, we determine r_m defined as the minimal number of rounds such that at least one complete monomial is present in each one of the 16 polynomials. We denote as complete monomial one monomial $x^e = \prod_{i=0}^{15} x_i^{e_i}$ such that for each $i \in [0, 15]$ it holds that $0 < e_i \leq p - 1$ (i.e., x^e depends on all the variables). Then, we add the number of rounds such that all possible degrees in one variable can be taken, in other words we add the number of rounds sufficient to wrap over p (note that the degree in x_i inside a monomial is always between 0 and $p - 1$ since $x_i^p = x_i$ over \mathbb{F}_p).

We next determine a bound on r_m . First, due to the Type-II Feistel structure, after 3 rounds not every input has an impact on the 16 outputs, which implies

$r_m > 3$. Then, we get an upper bound on r_m based on our experiments (the real value could be smaller, taking the upper bound is conservative). With $p = 3$ we obtain monomials depending on 10 variables in the polynomials in position 1, 2 and 3 and in positions 9, 10 and 11 by symmetry of the Type-II Feistel structure. These monomials contain all the variables from x_0 to x_7 and x_8 to x_{15} respectively. Calling X and Y monomials of this shape, we get that at round 5 there are terms of shape X+Y in the polynomials in positions 0, 1, 2, 3, 8, 9, 10 and 11, therefore giving complete monomials after passing through F due to the square operations. Since only half of the input goes through F at each round, one more round is needed to obtain these complete monomials in each position, therefore $r_m \leq 7$. When moving to $p = 2^7 - 1$ we can only observe more monomials (since all the ones with a coefficient multiple of 127 rather than 3 are canceled). Combined with the fact that the degree is at least 2^{2r-3} as shown before, we conclude that 5 extra rounds are sufficient to wrap around p and reach any degree in one variable. This gives us a bound of 12 rounds (equivalently, three steps S) to expect dense polynomials in the 16 positions.

5.3 Linearization Attack

Given a system of polynomial equations, one possible way to solve it is via the linearization technique which works by turning it into a system of linear equations and adding new variables that replace all the monomials in the system of degree larger than 1. The resulting linear system of equations can be solved using linear algebra if there are sufficiently many equations. Consider a system in x unknowns of degree limited by D , where the number of monomials $N(D, x)$ is given by $N(D, x) = \binom{D+x}{D}$ when $D < p$. The attack has a computational cost of $\mathcal{O}(N(D, x)^\omega)$ operations (for $2 < \omega \leq 3$), and a memory cost of $\mathcal{O}(N(D, x)^2)$ to store the linear equations. Depending on parameters' choices, the hybrid approach which combines exhaustive search with this resolution may lead to a reduced cost. Guessing $l < x$ variables leads to a complexity of:

$$\mathcal{O}(p^l \cdot N(D, x - l)^\omega) .$$

Case: $\tau = 0$ (no tweak). Since the key is composed of 16 \mathbb{F}_{2^7-1} -words, for any $l \in [0, 15]$ we computed that $p^l \cdot N(D, x - l)^\omega > 2^{112}$ occurs already for $D = 69$ (taking the conservative value of $\omega = 2$). Since the minimal degree follows 2^{2r-3} as shown previously (where $2^{2r-3} > 2^7 - 1$ for $r \geq 5$), and based on the density analysis just given, we can conclude that 3 steps S (equivalently, 12 rounds) are sufficient to prevent algebraic attacks based on linearization.

Case: $\tau \geq 1$ (related tweaks). The freedom of choosing the tweak(s) can be exploited to cancel some monomials whose coefficients depend on the tweak(s) or part of them. Similarly, the difference of two polynomials under related tweaks can be exploited to cancel monomials whose coefficients are independent of the tweaks. Moreover, the linear combinations of more polynomials under properly chosen related tweaks can be exploited to cancel monomials whose coefficients depend on the tweaks or part of them. In this last case, the attacker has to (i)

set up a system of equations in which the linear combinations of the coefficients of some monomials is set to zero, and (ii) solve it (e.g., via linearization or using Gröbner bases) to find the related tweaks that satisfy such conditions.

Obviously, this procedure is not free, since one has to solve equations in the tweak variables that are dense and of high (e.g., maximum) degree. Moreover, the non-linear tweak scheduling must be taken into account as well. Based on the analysis just given (and on the results presented in the full version for attacks based on Gröbner bases [47]), it would be infeasible for the attacker to solve a system of equations (in the tweaks instead of the plaintexts) that cover more than 12 rounds (or equivalently 3 steps) of the cipher once the tweak is fully absorbed (where we remind that 12 is the minimum number of rounds for achieving full diffusion in the interpolation polynomial). For this reason, we conjecture that $12 + 4 \cdot (\tau - 1) = 8 + 4 \cdot \tau$ extra rounds (or $2 + \tau$, extra steps, which means 3 or 4 for $\tau \in \{1, 2\}$) are sufficient for preventing related-tweak algebraic attacks. We note that this conjecture does not have to be tight for the security of **small-pSquare** to hold, since we need a larger number of extra steps to prevent related-tweak statistical attacks (respectively, 9 and 12 for $\tau \in \{1, 2\}$).

Note on Gröbner Bases Attacks. We recall that Gröbner bases based attacks reduce to linearization attacks when (i) the attacker aims to solve equations linking the plaintexts (and the tweaks) to the ciphertexts only, with the key as only variable, and (ii) the attacker can collect enough data for linearizing the system (i.e., the best scenario for the attacker). Hence, when analyzing the security of our scheme against such attacks in the full version [47], we only consider the case in which the system of equations is set up at round level.

6 Hardware Performance Evaluation of **small-pSquare**

In this section we evaluate the hardware cost and performance of the **small-pSquare** instance in comparison to respective implementations of the **SKINNY** lightweight tweakable block cipher [6]. Due to its simple and efficient design, **SKINNY** has gained remarkable popularity in recent years, both in academia and industry, and was selected as part of the ISO/IEC 18033-7:2022 standard for tweakable block ciphers. It has been designed with efficient application of side-channel countermeasures in mind, in particular masking, and is therefore ideally suited for our comparison [6]. Naturally, the general design strategy as well as the individual operations employed by the two ciphers (**small-pSquare** vs. **SKINNY**) are vastly different. At first sight, comparing two primitives with more differences than similarities may appear suboptimal to gain meaningful insights. However, in order to achieve a high level of cost-efficiency, lightweight TBCs necessarily need to be tailored to the amenities of their particular mathematical foundation. Hence, the stark differences between these primitives are a direct manifestation of their specialization to the finite fields they operate in. Only such a comparison can answer the question which mathematical setting (e.g., binary field vs. prime field masking) is preferable for constructing dedicated instances to maximize the efficiency vs. security tradeoff of protected TBC implementations.

Table 1: Cost and performance of round-based unprotected **SKINNY-128** and **small-pSquare** hardware implementations evaluated in TSMC 65 nm technology at typical operating conditions for 100 MHz and 250 MHz clock.

Cipher	Block Size	Key Size	Tweak Size	Freq. [MHz]	Crit. Path [ns]	Area [GE]	Power [mW]	Latency [cyc/enc]
SKINNY	128	128	0	100 250	1.877155	2450.75	0.3915	40
			128	100 250	1.812617	3396.00	0.5613	48
			256	100 250	1.905185	4353.00	0.7304	56
small-pSquare	112	112	0	100	9.777720	9684.75	1.3547	36
			112	100	9.970046	10798.75	1.5424	64
			224	100	9.937350	11989.50	1.6745	84
			0	250	3.945602	12407.75	1.7942	36
			112	250	3.971674	14716.50	2.1160	64
			224	250	3.972123	16034.00	2.2392	84

small-pSquare has been designed to offer competitive performances to common binary lightweight block ciphers when masking is applied. Nevertheless, we begin by comparing its critical path delay, area, power consumption (at 100 MHz operation) and encryption latency to **SKINNY-128** when both are implemented as unmasked round-based hardware circuits in Table 1. All values are post-synthesis results obtained using Synopsys Design Compiler Version O-2018.06-SP4 as a synthesis tool together with the TSMC 65 nm standard cell library at typical operating conditions for two different clock frequencies, 100 MHz and 250 MHz. The results show that regardless of the tweak size, unmasked **SKINNY-128** is significantly more efficient in terms of critical path delay, area footprint and power consumption when compared to unmasked **small-pSquare**. The encryption latency, which directly corresponds to the number of rounds, is slightly smaller for **small-pSquare** without tweak compared to **SKINNY-128** without tweak. However, for the tweakable variants it is larger in case of **small-pSquare**. We conclude that when unprotected, and hence for implementation settings where physical attacks are not a concern, **small-pSquare** is not fully competitive with binary lightweight ciphers in hardware. Yet, as mentioned before, this was not the primary goal of our design effort. Significantly better efficiency in unprotected hardware would have commanded different design choices that, in part, directly oppose to efficiency in masked representation.

We now focus on the more relevant comparison of secure higher-order masked hardware circuits. We have chosen to compare the **small-pSquare** version with $\tau = 1$ with **SKINNY-128-256** for 2 up to 4 shares (i.e., first- to third-order secure designs). **SKINNY-128-256** is the denotation of the **SKINNY** variant which receives a 128-bit plaintext, 128-bit key and 128-bit tweak (i.e., a 256-bit tweak) as inputs and computes 48 cipher rounds for one encryption or decryption. We recall that **small-pSquare** with $\tau = 1$ receives a 112-bit plaintext, 112-bit key and 112-bit tweak as inputs and computes 64 cipher rounds for one encryption or decryption. To put the difference of round numbers in perspective, remember that **small-pSquare** is a Type-II generalized Feistel design, i.e., each round updates only half of the state. We will see in the next results that with all other factors being equal, masked **small-pSquare** implementations generally require fewer clock

Table 2: Cost and performance comparison of masked SKINNY-128-256 and small-pSquare ($\tau = 1$) hardware implementations evaluated in TSMC 65 nm technology at typical operating conditions for 100 MHz and 250 MHz clock.

Cipher	Ref.	Par.	Frequ. [MHz]	d	Pip.	Crit. Path [ns]	Area [GE]	Power [mW]	Latency [cyc/enc]	Random [bit/cyc]
SKINNY-128-256		128	100 250	2		1.519177	19026.75	2.8547	432/1	128
				3		1.763878	38828.75	6.2545	432/1	384
				4		1.839592	65502.00	8.9225	432/1	768
		[58]	100 250	2	✓	1.566238	58475.50	13.6144	432/9	128
				3	✓	1.801272	94611.50	21.6698	432/9	384
				4	✓	1.882408	137625.50	30.8983	432/9	768
		32	100 250	2		1.743940	9274.50	1.0755	2160/1	32
				3		1.903482	15999.00	2.0608	2160/1	96
				4		1.823993	24442.00	8.2697	2160/1	192
			100 250	2	✓	1.885406	39016.25	9.1220	2160/9	32
				3	✓	1.943746	57757.00	13.4186	2160/9	96
				4	✓	1.909085	78243.00	17.9452	2160/9	192
small-pSquare	[85]	128	100 250	2		3.715469	18035.75	2.5276	288/1	32
				3		3.232731	28740.75	4.1347	288/1	96
				4		3.849724	41136.75	5.9918	288/1	192
			100	2		9.845555	21714.50	2.9370	128/1	84
				3		9.854049	41982.50	5.6533	128/1	210
				4		9.852280	62587.75	8.4822	128/1	504
		112	100	2	✓	9.852014	30730.25	4.4491	128/2	168
				3	✓	9.854022	65273.00	9.2764	128/2	420
				4	✓	9.853921	101168.00	14.3426	128/2	1008
			250	2		3.857306	29438.50	3.7809	128/1	84
				3		3.861372	52073.50	6.9574	128/1	210
				4		3.907730	78441.00	10.5274	128/1	504
	[this]		250	2	✓	3.852503	40414.75	5.5467	128/2	168
				3	✓	3.857475	77556.50	11.0357	128/2	420
				4	✓	3.859051	121589.25	17.4111	128/2	1008
			100	2		9.847473	15332.25	1.9296	256/1	42
				3		9.851035	27215.75	3.4077	256/1	105
				4		9.852068	39237.50	4.9897	256/1	252
		56	100	2	✓	9.848985	20735.75	2.8794	256/2	84
				3	✓	9.941982	39958.75	5.5274	256/2	210
				4	✓	9.851659	59404.75	8.2398	256/2	504
			250	2		3.855009	20471.00	2.4330	256/1	42
				3		3.858206	34485.25	4.2527	256/1	105
				4		3.859086	48511.25	6.1763	256/1	252
			250	2	✓	3.858980	26823.25	3.5092	256/2	84
				3	✓	3.858999	48147.50	6.5390	256/2	210
				4	✓	3.857775	72245.00	9.9639	256/2	504

Cipher = Evaluation target, either SKINNY-128-256 or small-pSquare ($\tau = 1$).
 Ref. = Reference, i.e., related publication, AGEMA is cited for automatically generated circuits.
 Par. = Parallelism, i.e., size of the state that is operated on in parallel measured in bits.
 Freq. = Synthesis frequency measured in Megahertz (MHz).
 d = Number of shares, resulting in security order $d - 1$.
 Pip. = Design is pipelined (✓) or not ().
 Crit. Path = Critical path of the synthesized circuit measured in nanoseconds (ns).
 Area = Area consumption of the synthesized circuit measured in gate equivalents (GE).
 Power = Power consumption of the synthesized circuit measured in milliwatts (mW).
 Latency = Latency of the synthesized circuit measured in clock cycles per encryption(s).
 Random = Fresh randomness consumption measured in bits per clock cycle.

cycles per encryption (sometimes significantly) than masked SKINNY-128-256 implementations, despite the larger number of rounds of the former design.

We have scanned the literature for publicly available securely masked SKINNY-128-256 implementations, with moderate success. The only higher-order masked hardware implementations of tweakable SKINNY-128 we could find have been published in conjunction with [85,78] as part of a study of the leakage resistance of Romulus and other Authenticated Encryption with Associated Data (AEAD) schemes that made it into the finals of NIST’s Lightweight Crypto Competition. The concrete implementation is hence of the SKINNY-128-384+ variant which is used in Romulus and publicly available on GitHub.⁸ Please note that the authors have not verified its security properties experimentally. However, it is based on the trivially composable HPC2 masking scheme [26] which eases the extension of gadget security to full-implementation security. We have modified this implementation slightly to make it compute SKINNY-128-256 instead of SKINNY-128-384+, which commanded small changes to the round numbers and tweak schedule. Since this concrete implementation uses a specific implementation of the SKINNY 8-bit S-box that is tailored towards a certain set of optimization goals, we also wanted to include other, more general, masked SKINNY-128 implementations in our comparison. To this end we have employed the Automated Generation of Masked Hardware (AGEMA) tool published at TCHES 2022 [58] which is able to turn unprotected hardware implementations of cryptographic primitives automatically into securely masked equivalents. We have utilized the tool to autonomously generate masked implementations of SKINNY-128-256 from the source code for unprotected hardware circuits provided by the SKINNY authors on its website as source material.⁹ In particular, we translated both, round-based and 32-bit serialized implementations into their masked equivalents based on the HPC2 masking scheme using the Naive processing method (see [58]), as it led to the most suitable results for a comparison. We further generated both pipelined and non-pipelined masked circuits. Given this collection of securely masked SKINNY-128-256 circuits we are now equipped for an in-depth cost and performance comparison to our prime-field TBC.

Analogously to the selected implementations of SKINNY, we created round-based and half-round-based masked hardware circuits of **small-pSquare**. While these implementations operate on the full state (112 bits) and half the state (56 bits) in parallel, respectively, non-linear operations are only applied to 56 and 28 bits in parallel respectively due to the Feistel structure. Hence, the half-round-based implementations compute only one F-function on a 28-bit input at a time, resulting in a similar serialization level (28 vs. 32) compared to the SKINNY equivalent. The circuits are based on the secure and composable prime-field squaring gadgets introduced in [27]. In fact, we even optimized the 4-share gadget in a way that it only needs a single register stage, using similar optimization strategies as for the 2-share and 3-share case presented by the authors of [27]. The pseudo-code for the gadget is included in the full version of this paper [47]. The resulting comparison is presented in Table 2. All results are based on post-synthesis estimations obtained using Synopsys Design Compiler and TSMC 65 nm technology

⁸ https://github.com/uclcrypto/aead_modes_levelled_hw

⁹ <https://sites.google.com/site/skinnycipher/implementation>

at typical operating conditions for two different frequencies, 100 MHz and 250 MHz. The resulting figures for the **SKINNY-128-256** circuits are identical for both frequencies due to the short critical path length. The full version of this paper additionally considers the more extreme cases of 500 MHz and 1000 MHz operation [47].¹⁰ Based on this collection of cost and performance results, we conclude that **small-pSquare** is indeed able to compete with **SKINNY-128-256** when masked in hardware, especially at lower frequencies. The automatically generated hardware circuits of **SKINNY** need a rather large amount of cycles per round (regardless of the frequency) and are costly when pipelined. At the target frequencies considered in the table (≤ 250 MHz) **SKINNY-128-256** is only consistently cheaper in terms of area when non-pipelined serialized implementations are compared. Yet, this advantage in area footprint comes at a steep price, as the encryption latency is larger by a considerable factor. Overall, considering the area/power consumption and the latency together, **small-pSquare** often appears preferable. This changes when very high operating frequencies are needed. Then **SKINNY** becomes preferable, as shown in the full version [47]. In summary, our results show that **small-pSquare** is indeed capable of providing competitive cost and performance results in the envisioned application settings, and even outperforms its competitor consistently when the frequency is sufficiently low.

Decryption. Our comparison focuses on encryption-only circuits. However, adding capability for decryption is trivial for **small-pSquare** due to the Feistel structure. The additional cost for multiplexing between addition and subtraction of **F**-function results for the encryption and decryption process falls in the range of a few percent (depending on masking order and parallelization). Adding decryption capability to the **SKINNY** circuits typically requires twice the area.

Comparison to AES-prime. We note that **small-pSquare** is also significantly more efficient in hardware compared to **AES-prime** which has been introduced at Eurocrypt 2023 as the first example of a dedicated cipher for prime-field masking [67]. It shares the same block and key size as **small-pSquare**, but is not tweakable. Compared to **AES-prime** our unmasked **small-pSquare** with $\tau = 0$ is at least 3 times smaller. More importantly, masked **small-pSquare** with $\tau = 1$ is on average (over the number of shares) 5 times smaller compared to masked **AES-prime** despite the additional tweak input [67]. This implies that masked **small-pSquare** with $\tau = 0$ requires a more than 5 times smaller area footprint while also executing in fewer clock cycles under the same frequency, constituting a very notable improvement over the state of the art. In addition, we recall that all variants of **small-pSquare** enable efficient decryption, while **AES-prime** does not.

Area and power consumption. It is clear that the comparison of the area and power consumption in Table 2 is affected by the different block, key and tweak

¹⁰ More “extreme”, because cryptographic co-processors manufactured in 65 nm technology are rarely clocked at such high frequencies. This is evident for example in research ASICs manufactured in such technology nodes, as reported in <http://asic.ethz.ch/technologies/65.html>. Furthermore, the vast majority of common criteria certified co-processors protected against side-channel attacks do not exceed 200-300 MHz operation (<https://www.commoncriteriaportal.org/products>).

sizes of the analyzed primitives. Since all those size parameters are smaller for the chosen prime-field design, this fact may tilt the comparison of the implementation size to **small-pSquare**'s advantage. However, we argue that even if normalizing all area and power figures related to **small-pSquare** artificially by, for example, multiplying them with the corresponding size difference factor, namely $\frac{128}{112} = \frac{8}{7}$, our conclusions would not change drastically. In order to avoid any confusion we have *not* applied any artificial normalization of our results. We would also like to mention that for cases where 112 bits of security are insufficient, increasing the security level of **small-pSquare** at low additional cost is possible using the trick employed by the PRINCE block cipher with whitening keys [22].

Latency vs. frequency. **small-pSquare** naturally allows to trade latency in cycles for frequency in Megahertz and vice versa. The considered masked **SKINNY-128-256** implementations offer less flexibility. They always have a very short critical path but require a larger number of cycles due to the type of masked gadgets that are used. In fact, bit-wise masking, where each binary two-input non-linear gate (e.g., AND, OR, NAND, NOR) is individually replaced by a masked gadget equivalent, enables high frequency operation but requires many register stages to uphold masking security. This is because any secure masked hardware gadget computing a non-linear operation requires at least one register stage (attempts to improve this are usually based on additional specialized hardware assumptions [75,70]). Introducing a register stage for each atomic bit-level gate entails a high overhead in latency of the implementation, but also in area. While this makes masked **SKINNY-128-256** well-suited for high frequency operation, it limits its performance in lower frequency and low-latency applications.

Mild additional constraints. We note that **small-pSquare** comes with a few additional constraints due to the fact that it operates in a prime field while data is usually encoded in a binary manner. It is however pretty simple to convert a vector of prime field elements into a sequence of bits, by just viewing it as a representation of an integer in basis p . For **small-pSquare**, the maximum value is worth $127^{16} - 1$ which can represent 111-bit values (yielding a one-bit loss in the conversion). Similarly, masking **small-pSquare** requires to generate uniformly random prime numbers. Rejection sampling is a viable method (with probability $1/127$ to reject a value). Using a PRNG that natively operates in \mathbb{F}_p is an alternative. Eventually, **small-pSquare** would be best integrated in a leakage-resistant mode of operation, which should not raise specific problems since TBC-based constructions like [9] or [78] are field-agnostic. Overall, none of these minor caveats is expected to bring significant overheads.

7 Side-Channel Security Assessment of **small-pSquare**

Finally, we evaluate the security of our masked implementations. In particular, we experimentally assess and compare the side-channel resistance of masked **small-pSquare** ($\tau = 1$) and **SKINNY-128-256** implementations by measuring their power consumption on an FPGA device and trying to infer the secret key from

Table 3: Comparative side-channel evaluation of pipelined masked SKINNY-128-256 and small-pSquare hardware implementations.

Cipher	Par.	CPR	d	mean SNR	median SNR	TVLA det. compl.	SASCA compl.
SKINNY-128-256	128	9	2	0.0023	0.0021	71 000	25 000
			3	0.0015	0.0019	811 000	362 000
			4	0.0013	0.0010	57 000 000	29 832 000
	32	45	2	0.0064	0.0048	52 000	6 000
			3	0.0026	0.0028	680 000	157 000
			4	0.0020	0.0016	48 000 000	17 169 000
small-pSquare	112	2	2	0.0021	0.0019	321 000	213 000
			3	0.0032	0.0013	8 040 000	4 002 000
			4	0.0016	0.0011	> 100 000 000	> 100 000 000
	56	4	2	0.0073	0.0031	238 000	45 000
			3	0.0030	0.0025	7 040 000	1 754 000
			4	0.0018	0.0020	> 100 000 000	> 100 000 000

Cipher = Evaluation target, either SKINNY-128-256 or small-pSquare ($\tau = 1$).

Par. = Parallelism, i.e., size of the state that is operated on in parallel measured in bits.

CPR = Cycles per round, i.e., latency of one round function computation measured in cycles.

d = Number of shares, resulting in security order $d - 1$.

mean SNR = Mean maximum SNR of all S-box/Squaring input shares in the first round.

median SNR = Median maximum SNR of all S-box/Squaring input shares in the first round.

TVLA det. compl. = Minimum number of traces to surpass the TVLA detection threshold.

SASCA compl. = Minimum number of traces to achieve key rank 1 in a SASCA key recovery.

their side-channel leakage. We focus on pipelined implementations of small-pSquare and SKINNY-128-256 for 2, 3 and 4 shares. The serialized circuits constitute a scenario with lower noise, while the parallel ones help to show the differences at slightly higher noise levels, although their side-channel Signal-to-Noise Ratios (SNRs) are not drastically different. The full version of this paper additionally contains the evaluation results for the small-pSquare versions with extra register stages to enable larger maximum frequencies [47].

Setup. For our experiments we use a SAKURA-G FPGA board which houses two Spartan-6 FPGAs serving as controller and device under test, respectively. All designs are operated at 6 MHz clock frequency (for low noise) and their power consumption is measured using a PicoScope 5244D digital sampling oscilloscope at 250 MS/s sampling rate with 12-bit vertical resolution through a Tektronix CT-1 current probe (bandwidth of up to 1 GHz) placed in the power supply path of the target FPGA. Xilinx ISE Version 14.7 is used to synthesize the circuits, with default parameters except the `-keep_hierarchy` attribute set to `yes`.

Table 3 summarizes the evaluation results collected for the 12 different masked implementations, $6 \times$ SKINNY-128-256 and $6 \times$ small-pSquare. It is apparent from the mean and median side-channel SNRs computed over all first-round 8-bit S-box (SKINNY-128-256) input shares or 7-bit Squaring (small-pSquare) input shares, that the quality of observations an adversary can make of individual words processed in the circuits is quite similar in both cases (binary-field or prime-field cipher). Thus, the noise levels are not expected to significantly impact the following investigation. We have plotted one set of evaluation results in Figure 4 for the concrete example of serialized pipelined implementations with 3 shares. For all 2-, 3- and 4-share implementations respectively, we have first measured 1 million, 10 million and 100 million traces, in a randomly interleaved sequence of measurements for fixed and for random inputs, according to

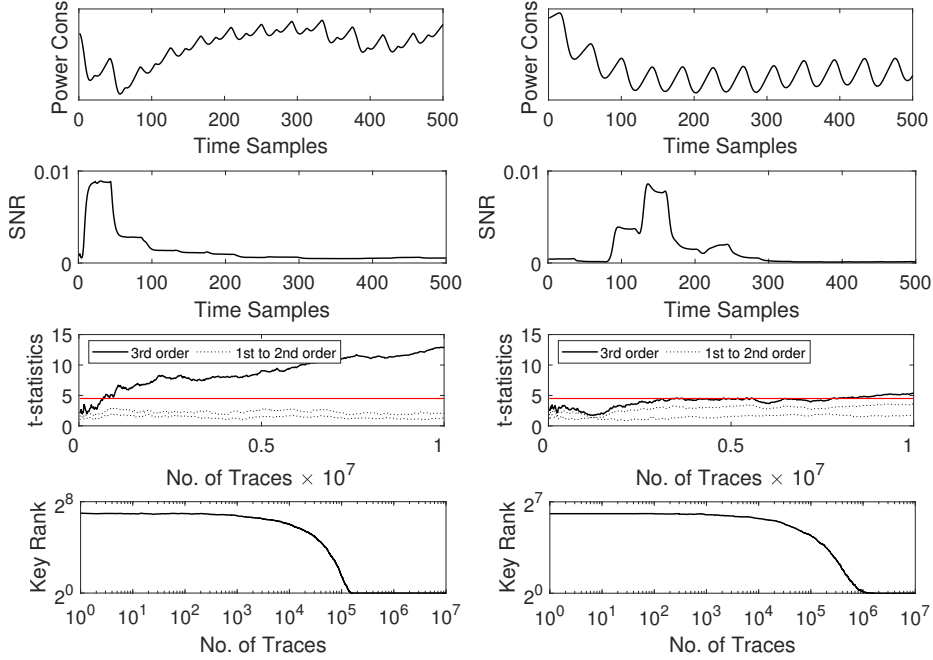


Figure 4: Exemplary SCA results of serialized second-order masked SKINNY-128-256 (left) and small-pSquare (right) hardware implementations. From top to bottom: Sample traces, SNRs (1M traces), Fixed-vs-random t-tests (10M traces), Profiled SASCA (1M profiling, 10M attack traces).

the Test Vector Leakage Assessment (TVLA) methodology [77]. The resulting non-specific t-test results are illustrated in the third row of Figure 4. The implementations satisfy the expected statistical security order in the experiments, as the smallest moment where leakage is detected is equal to the number of shares. The same holds for all evaluated circuits in Table 3. As a next step we performed exemplary key recovery attacks on the most leaking (highest SNR) 8-bit or 7-bit word respectively of the state. In order to extract the most information from the traces and reduce the effective noise level we have employed a profiled horizontal Soft-Analytical Side-Channel Attack (SASCA) [86]. In a first step all relevant intermediate values are profiled over multiple clock cycles to obtain multivariate templates. Next, a Linear Discriminant Analysis (LDA) is used to perform a linear dimensionality reduction which maximizes class separation on the profiling traces (always 1M) [81]. Finally, on the distinct attack trace set, likelihoods for all intermediate values and corresponding templates are collected separately, before a discrete probability distribution of the secret value is derived using belief propagation inside a SASCA tree graph that contains multiple intermediate computation stages of the masked S-box or squaring. These procedures are readily implemented in the publicly available **SCALib** library [25]. We then estimate the average rank of the correct key (over 1 000 iterations) with the probabilities obtained from all the attack traces. The results of that proce-

ture plotted over the number of attack traces are shown in the bottom row of Figure 4. In both the TVLA and the SASCA results it is apparent that, despite similar SNR values, successful leakage detection and key recovery require consistently significantly more traces (higher data complexity) on the **small-pSquare** compared to the **SKINNY** circuits, regardless of the concrete implementation chosen. This advantage of prime-field masking can be attributed to the “algebraic incompatibility” between physical leakage and recombination function to compute the secret from its shares. It can be observed that in case of attacks on 3- and 4-share implementations the advantage of **small-pSquare** is around or above one order of magnitude (slightly less in case of 2 shares). Furthermore no leakage detection or sophisticated key recovery attack succeeded on any of the 4-share implementations of **small-pSquare** using 100 million traces. Despite their empirical nature, we believe these results clearly emphasize the interest of efficient TBCs dedicated for prime-field masking based on established design principles (e.g., Feistel structures) while also tailoring the design to specific advantages that a given mathematical structure can lead to (e.g., using squaring as source of non-linearity in prime fields to exploit their efficient masked gadgets).

8 Summary and Open Problems

In this paper, we proposed both the FPM family of ciphers that leverages a generalized Feistel structure for prime masking and easy integration in leakage-resistant modes of operation, and the **small-pSquare** instance that is tailored for hardware implementations (due to its small prime) and exploits recent advances for masked squaring gadgets. Combining a hardware performance evaluation with an initial side-channel security assessment allows us to put forward the interest of this approach. **small-pSquare** protected with prime masking shows significantly improved (side-channel) security vs. performance tradeoffs compared to **SKINNY** protected with Boolean masking. Besides their concrete interest, we believe our investigations uncover new design principles for side-channel resistant implementations, leading to new challenges for further research.

Starting from more specific questions, the mathematical and physical security evaluation of **small-pSquare** is, as usual for new ciphers, a natural direction for deeper analyzes. Given the breadth of the FPM family, investigating other instances would be interesting as well. For example, a **mid-pSquare** instance with $p = 2^{31} - 1$ would be particularly well-suited for software implementations for which masking is known to be difficult to implement due to a lack of noise [5,24]. Such an instance could for example be based on the high-level structure depicted in the left part of Figure 2 combined with the candidate **F** function given in Figure 5. For modes of operation where having efficient inverses is not critical, it could also be possible to replace the generalized Feistel structure of Section 2.2 by an SPN one. More generally, the use of prime masking raises important theoretical questions regarding security proofs. For example, while the seed results of Dziembowski et al. provide a rationale for prime masking [41], the understanding of this approach is still far from the one of Boolean masking. Typical open problems in this respect are to improve the tightness of the security proofs and

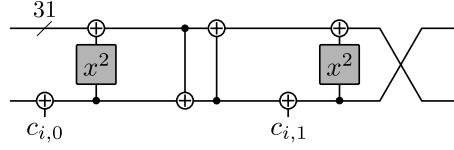


Figure 5: Candidate F-function for a mid-pSquare instance.

to better formalize the intuition of “algebraic incompatibility” that makes prime computations less sensitive to a lack of noise than Boolean masking.

The source code for all our small-pSquare implementations is publicly available here: <https://github.com/uclcrypto/small-pSquare>

Acknowledgments. Lorenzo Grassi was supported by the German Research Foundation (DFG) within the framework of the Excellence Strategy of the Federal Government and the States – EXC 2092 CaSa – 39078197. Pierrick Méaux was supported by the ERC Advanced Grant 787390. François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in parts by the ERC Advanced Grant 101096871. Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the ERC. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel Structures for MPC, and More. In *ESORICS*, 2019.
2. Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *ASIACRYPT*, 2016.
3. Abdelrahman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. *IACR Trans. Symmetric Cryptol.*, 2020(3).
4. Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. Theory and Practice of a Leakage Resilient Masking Scheme. In *ASIACRYPT*, 2012.
5. Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal Side-Channel Attacks and Countermeasures on the ISW Masking Scheme. In *CHES*, 2016.
6. C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *CRYPTO*, 2016.
7. D. Bellizia, O. Bronchain, G. Cassiers, V. Grosso, C. Guo, C. Momin, O. Pereira, T. Peters, and F.-X. Standaert. Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography - A Practical Guide Through the Leakage-Resistance Jungle. In *CRYPTO*, 2020.
8. Thierry P. Berger, Marine Minier, and Gaël Thomas. Extended Generalized Feistel Networks Using Matrix Representation. In *Selected Areas in Cryptography*, 2013.
9. Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Tedt, a leakage-resist AEAD mode for high physical security applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1).

10. Francesco Berti, Chun Guo, Thomas Peters, and François-Xavier Standaert. Efficient Leakage-Resilient MACs Without Idealized Assumptions. In *ASIACRYPT*, 2021.
11. Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On Leakage-Resilient Authenticated Encryption with Decryption Leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3).
12. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In *EUROCRYPT*, 2008.
13. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In *SAC*, 2011.
14. Tim Beyne and Yu Long Chen. Provably Secure Reflection Ciphers. In *CRYPTO*, 2022.
15. Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptol.*, 7(4), 1994.
16. Eli Biham. A Fast New DES Implementation in Software. In *FSE*, 1997.
17. Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *EUROCRYPT*, 1999.
18. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptol.*, 4(1), 1991.
19. Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.
20. Alex Biryukov and David A. Wagner. Slide Attacks. In *FSE*, 1999.
21. Alex Biryukov and David A. Wagner. Advanced Slide Attacks. In *EUROCRYPT*, 2000.
22. J. Borghoff, A. Canteaut, T. Güneysu, E. Bilge Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçın. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications. In *ASIACRYPT*, 2012.
23. D. Bozilov, M. Eichlseder, M. Knezevic, B. Lambin, G. Leander, T. Moos, V. Nikov, S. Rasoolzadeh, Y. Todo, and F. Wiemer. PRINCEv2 - More Security for (Almost) No Overhead. In *Selected Areas in Cryptography*, 2020.
24. Olivier Bronchain and François-Xavier Standaert. Breaking Masked Implementations with Many Shares on 32-bit Software Platforms or When the Security Order Does Not Matter. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3).
25. Gaëtan Cassiers and Olivier Bronchain. Scalib: A side-channel analysis library. *J. Open Source Softw.*, 8(86), 2023.
26. Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware Private Circuits: From Trivial Composition to Full Verification. *IEEE Trans. Computers*, 70(10), 2021.
27. Gaëtan Cassiers, Loïc Masure, Charles Momin, Thorben Moos, and François-Xavier Standaert. Prime-Field Masking in Hardware and its Soundness against Low-Noise SCA Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2).
28. Victor Cauchois, Clément Gomez, and Gaël Thomas. General Diffusion Analysis: How to Find Optimal Permutations for Generalized Type-II Feistel Schemes. *IACR Trans. Symmetric Cryptol.*, 2019(1).
29. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO*, 1999.
30. Jihoon Cho, Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Joohee Lee, Jooyoung Lee, Dukjae Moon, and Hyojin Yoon. Transciphering Framework for Approximate Homomorphic Encryption. In *ASIACRYPT*, 2021.

31. Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *EUROCRYPT*, 2000.
32. Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. Nessie Proposal: NOEKEON. First Open NESSIE Workshop, 2000.
33. Joan Daemen and Vincent Rijmen. The Wide Trail Design Strategy. In *Cryptography and Coding, 8th IMA International Conference 2001*, 2001.
34. Joan Daemen and Vincent Rijmen. Security of a Wide Trail Design. In *INDOCRYPT*, 2002.
35. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.*, 34(3), 2021.
36. Christoph Dobraunig, Lorenzo Grassi, Anna Guinet, and Daniël Kuijsters. Cimin-ion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields. In *EUROCRYPT*, 2021.
37. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. In *EUROCRYPT*, 2014.
38. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device. In *EUROCRYPT*, 2015.
39. Sébastien Duval and Gaëtan Leurent. MDS Matrices with Lightweight Circuits. *IACR Trans. Symmetric Cryptol.*, 2018(2).
40. Stefan Dziembowski and Sebastian Faust. Leakage-Resilient Cryptography from the Inner-Product Extractor. In *ASIACRYPT*, 2011.
41. Stefan Dziembowski, Sebastian Faust, and Maciej Skórski. Optimal Amplification of Noisy Leakages. In *TCC (A2)*, 2016.
42. Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine Masking against Higher-Order Side Channel Analysis. In *Selected Areas in Cryptography*, 2010.
43. Jovan Dj. Golic and Christophe Tymen. Multiplicative Masking and Power Analysis of AES. In *CHES*, 2002.
44. Dahmun Goudarzi and Matthieu Rivain. How Fast Can Higher-Order Masking Be in Software? In *EUROCRYPT*, 2017.
45. Lorenzo Grassi. On Generalizations of the Lai-Massey Scheme. Cryptology ePrint Archive, Paper 2022/1245, 2022. <https://eprint.iacr.org/2022/1245>.
46. Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In *EUROCRYPT*, 2020.
47. Lorenzo Grassi, Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert. Generalized Feistel Ciphers for Efficient Prime Field Masking - Full Version. Cryptology ePrint Archive, 2024.
48. Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. Subspace Trail Cryptanalysis and its Applications to AES. *IACR Trans. Symmetric Cryptol.*, 2016(2).
49. Hannes Groß, Stefan Mangard, and Thomas Korak. An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order. In *CT-RSA*, 2017.
50. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations. In *FSE*, 2014.
51. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In *CHES*, 2011.

52. Viet Tung Hoang and Phillip Rogaway. On Generalized Feistel Networks. In *CRYPTO*, 2010.
53. D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *CHES*, 2006.
54. Yuval Ishai, Amit Sahai, and David A. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, 2003.
55. Thomas Jakobsen and Lars R. Knudsen. The Interpolation Attack on Block Ciphers. In *FSE*, 1997.
56. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In *ASIACRYPT*, 2014.
57. Stéphanie Kerckhof, François Durvaux, Cédric Hocquet, David Bol, and François-Xavier Standaert. Towards Green Cryptography: A Comparison of Lightweight Ciphers from the Energy Viewpoint. In *CHES*, 2012.
58. David Knichel, Amir Moradi, Nicolai Müller, and Pascal Sasdrich. Automated Generation of Masked Hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1).
59. Lars R. Knudsen. Truncated and Higher Order Differentials. In *FSE*, 1994.
60. Thorsten Kranz, Gregor Leander, Ko Stoffelen, and Friedrich Wiemer. Shorter Linear Straight-Line Programs for MDS Matrices. *IACR Trans. Symmetric Cryptol.*, 2017(4).
61. Xuejia Lai. Higher Order Derivatives and Differential Cryptanalysis. In *Communications and Cryptography: Two Sides of One Tapestry*, 1994.
62. Xuejia Lai and James L. Massey. A Proposal for a New Block Encryption Standard. In *EUROCRYPT*, 1990.
63. Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack. In *CRYPTO*, 2011.
64. Gregor Leander, Brice Minaud, and Sondre Rønjom. A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro. In *EUROCRYPT*, 2015.
65. Chaoyun Li and Qingju Wang. Design of Lightweight Linear Diffusion Layers from Near-MDS Matrices. *IACR Trans. Symmetric Cryptol.*, 2017(1).
66. Meicheng Liu and Siang Meng Sim. Lightweight MDS Generalized Circulant Matrices. In *FSE*, 2016.
67. Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert. Effective and Efficient Masking with Low Noise Using Small-Mersenne-Prime Ciphers. In *EUROCRYPT*, 2023.
68. Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In *EUROCRYPT*, 1993.
69. Thorben Moos, Amir Moradi, and Bastian Richter. Static power side-channel analysis of a threshold implementation prototype chip. In *DATE*, 2017.
70. Rishub Nagpal, Barbara Gigerl, Robert Primas, and Stefan Mangard. Riding the Waves Towards Generic Single-Cycle Masking in Hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4).
71. Kaisa Nyberg. Generalized Feistel Networks. In *ASIACRYPT*, 1996.
72. Santos Merino Del Pozo, François-Xavier Standaert, Dina Kamel, and Amir Moradi. Side-channel attacks from static power: when should we care? In *DATE*, 2015.
73. Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks: A Formal Security Proof. In *EUROCRYPT*, 2013.

74. Thomas Roche and Emmanuel Prouff. Higher-order glitch free implementation of the AES using Secure Multi-Party Computation protocols - Extended version. *J. Cryptogr. Eng.*, 2(2), 2012.
75. Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E. Marson. Low-Latency Hardware Masking with Application to AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2).
76. Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In *CHES*, 2005.
77. Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES*, 2015.
78. Yaobin Shen, Thomas Peters, François-Xavier Standaert, Gaëtan Cassiers, and Corentin Verhamme. Triplex: an Efficient and One-Pass Leakage-Resistant Mode of Operation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4).
79. Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-Bit Blockcipher CLEFIA. In *FSE*, 2007.
80. François-Xavier Standaert. How (Not) to Use Welch's T-Test in Side-Channel Security Evaluations. In *CARDIS*, 2018.
81. François-Xavier Standaert and Cédric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In *CHES*, 2008.
82. François-Xavier Standaert, Gilles Piret, Gaël Rouvroy, and Jean-Jacques Quisquater. FPGA implementations of the ICEBERG block cipher. *Integr.*, 40(1), 2007.
83. Tomoyasu Suzaki and Kazuhiko Minematsu. Improving the Generalized Feistel. In *FSE*, 2010.
84. Serge Vaudenay. On the Lai-Massey Scheme. In *ASIACRYPT*, 1999.
85. Corentin Verhamme, Gaëtan Cassiers, and François-Xavier Standaert. Analyzing the Leakage Resistance of the NIST's Lightweight Crypto Competition's Finalists. In *CARDIS*, 2022.
86. Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft Analytical Side-Channel Attacks. In *ASIACRYPT*, 2014.
87. David A. Wagner. The Boomerang Attack. In *FSE*, 1999.
88. Lei Wang, Jian Guo, Guoyan Zhang, Jingyuan Zhao, and Dawu Gu. How to Build Fully Secure Tweakable Blockciphers from Classical Blockciphers. In *ASIACRYPT*, 2016.
89. Weijia Wang, Pierrick Méaux, Gaëtan Cassiers, and François-Xavier Standaert. Efficient and Private Computations with Code-Based Masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2).
90. Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In *CRYPTO*, 1989.