IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2025, No. 4, pp. 486–519. DOI:10.46586/tches.v2025.i4.486-519

mid-pSquare: Leveraging the Strong Side-Channel Security of Prime-Field Masking in Software

Brieuc Balon[1], Lorenzo Grassi[2], Pierrick Méaux[3], Thorben Moos[1], François-Xavier Standaert[1] and Matthias Johann Steiner[4]

[1] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium
firstname.lastname@uclouvain.be
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
l.grassi@tue.nl
[3] Luxembourg University, Esch-sur-Alzette, Luxembourg
pierrick.meaux@uni.lu
[4] Alpen-Adria-Universität Klagenfurt, Klagenfurt am Wörthersee, Austria
mattsteiner@edu.aau.at

Abstract. Efficiently protecting embedded *software* implementations of standard symmetric cryptographic primitives against side-channel attacks has been shown to be a considerable challenge in practice. This is, in part, due to the most natural countermeasure for such ciphers, namely Boolean masking, not amplifying security well in the absence of sufficient physical noise in the measurements. So-called prime-field masking has been demonstrated to provide improved theoretical guarantees in this context, and the Feistel for Prime Masking (FPM) family of Tweakable Block Ciphers (TBCs) has been recently introduced by Grassi *et al.* (Eurocrypt'24) to efficiently leverage these advantages. However, it was so far only instantiated for and empirically evaluated in a hardware implementation context, by using a small (7-bit) prime modulus.

In this paper, we build on the theoretical incentive to increase the prime field size to obtain improved side-channel (Faust *et al.*, Eurocrypt'24) and fault (Moos *et al.*, CHES'24) resistance, as well as on the practical incentive to instantiate an FPM instance with optimized performance on 32-bit software platforms. We introduce mid-pSquare for this purpose, a lightweight TBC operating over a 31-bit Mersenne prime field. We first provide an in-depth black-box security analysis with a particular focus on algebraic attacks – which, contrary to the cryptanalysis of instances over smaller primes, are more powerful than statistical ones in our setting. We also design a strong tweak schedule to account for potential related-tweak algebraic attacks which, so far, are almost unknown in the literature. We then demonstrate that mid-pSquare implementations deliver very competitive performance results on the target platform compared to analogous binary TBCs regardless of masked or unmasked implementation (we use fix-sliced SKINNY for our comparisons). Finally, we experimentally establish the side-channel security improvements that masked mid-pSquare can lead to, reaching unmatched resistance to profiled horizontal attacks on lightweight 32-bit processors (ARM Cortex-M4).

Keywords: Side-Channel Attacks · Prime Ciphers · Software Masking

1 Introduction

Differential Power Analysis (DPA) is a very powerful attack technique that reduces the secrecy of (e.g., a block cipher) secret key exponentially in the number of measurements obtained by the adversary [KJJ99]. Masking is one of the few countermeasures that

Received: 2025-04-15 Accepted: 2025-06-15 Published: 2025-09-05

can theoretically mitigate this exponential security decrease, by itself amplifying the noise in the measurements exponentially in its number of shares [CJRR99]. Such a noise increase can in turn improve side-channel security exponentially in this (number of shares) parameter [PR13, DDF14, DFS15]. Yet, this amplification typically requires that implementations are sufficiently noisy in the first place, a condition that has been shown to hardly be met by software executed on 32-bit processors, which are popular targets for lightweight embedded security [BCPZ16, BS21]. Concretely, these results show that without additional (hardware) countermeasures, ensuring side-channel security in software implementations is doomed to require a (prohibitively) high number of shares, leading to low efficiency.

However, recent works on masking over prime-order fields hint that this unsatisfactory situation is not irremediable. In particular, and building on the seminal theoretical advances of Dziembowski et al. [DFS16], it has been shown that:

- Masking in prime-order fields can provide strong side-channel security even in the context of (very) low-noise implementations [MMMS23].
- Dedicated lightweight ciphers to facilitate prime-field masking can be very efficient [GMM$^+$24], and leverage secure (masked) squaring gadgets [CMM$^+$23].
- Prime-field masking brings significant advantages for fault resistance [MSS24].
- Increasing the field/word size is highly beneficial for amplifying both side-channel resistance [FMM$^+$24] and fault resistance [MSS24] even further.

Yet, for now, the only ciphers specialized for prime masking, AES-prime [MMMS23] and small-pSquare [GMM$^+$24] (an instance of the Feistel for Prime Masking (FPM) family), have been tailored to hardware implementations and use a small (7-bit) prime modulus. Since parallelizing computations over the word size of a processor is not directly feasible for prime-field arithmetic, such primitives are not ideally suited to demonstrate the potential of prime masking on *software* platforms, where low noise levels are most commonly observed.

Our Contributions. In this work, we propose mid-pSquare, a new instance of the FPM family of Tweakable Block Ciphers (TBCs), and analyze its black-box and side-channel security. In both cases, the main challenges relate to the larger (31-bit) prime modulus it uses to reach high security and efficiency on software platforms. From the (black-box) cryptanalysis viewpoint, it implies that the best attack vectors against the design are now algebraic ones (in contrast to statistical ones for small-pSquare). Hence, we focus on this type of cryptanalysis techniques. We believe that the depth of our algebraic analysis can provide new insights for family instances over larger fields even beyond our concrete design. In particular, while statistical related-tweak attacks are well-known in the cryptanalysis literature, algebraic ones are almost non-existent. Thus, presenting the challenge of considering such attacks even before their concrete application to any primitives. We address this uncertainty by designing an algebraically strong yet efficient tweak schedule, which is another main difference to small-pSquare.

From the side-channel analysis viewpoint, we are able to provide evidence for the strongly superior security of masked mid-pSquare implementations on an ARM Cortex-M4 device, by comparing it to (masked) fix-sliced SKINNY [AP21], which is a state-of-the-art lightweight TBC [BJK$^+$16]. Such an analysis does not come without hurdles, since directly profiling 32-bit operations and leveraging 32-bit key guesses in higher-order side-channel attacks with a large number of measurements is computationally prohibitive [YK21, CDSU23]. Hence, we provide a comprehensive discussion of attack and extrapolation strategies in order to reach sound conclusions. To the best of our knowledge, this is the first work that demonstrates high security against strong profiled horizontal/multivariate attacks in *software* with a number of shares that leads to affordable performance figures.

For completeness, we also discuss in Appendix B the orthogonal issue of transitional leakages from microarchitectural overwriting [CGP+12, BGG+14], and show both information theoretically and in practice, that it is less of a concern for prime-field masking.

In summary, we find that mid-pSquare is very easy to implement and mask, highly efficient on various platforms (especially with single-cycle 32-bit unsigned multiplication instructions available) and, for comparable if not better performance figures, provides side-channel security guarantees that can be orders of magnitude larger than for comparable binary TBCs like SKINNY. Given that prime-field masking also provides concrete advantages for fault resistance, especially for larger field sizes [MSS24], we conclude that mid-pSquare improves the state of the art for multiple relevant aspects of secure software implementations.

Cautionary note #1: Why prime-field masking? The aforelisted references established prime-field masking as an interesting alternative for securing embedded software devices against side-channel attacks in low-noise contexts. Intuitively, the important advantage they bring over Boolean masking is a lower "algebraic compatibility" with the leakage functions usually observed in practice, like the Hamming weight (HW) function [MOP07] or weighted sums of bits [SLP05]. This is because without noise, knowing for example the Hamming weight of Boolean shares leads to significant information about the masked secret (parity of its HW), independent of the number of shares. Leaking the Least Significant Bit (LSB) of d Boolean shares directly provides the LSB of the secret. By contrast, Dziembowski et al. showed that with prime-field encodings, security amplification (in the number of shares) only requires the leakage function to be non-injective, independent of its shape [DFS16]. Leaking the LSB of d prime-field additive shares does not directly translate into a bit of the secret.[1] Concretely, the important implication of this result, confirmed experimentally by the aforelisted references on prime-field masking, is that the increased security of such encodings indeed originates from their algebraic nature and is not caused by a variation of the Signal-to-Noise Ratio (SNR) of the side channel [Man04], which remains largely unaffected by moving from Boolean to prime-field masking.

Cautionary note #2: Why SKINNY? In order to exhibit the good physical security vs. performance tradeoff that mid-pSquare enables, we need to compare it with a competitor. Given that small-pSquare (the original FPM instance from [GMM+24]) was designed with hardware implementations in mind, it is not expected to perform well in 32-bit software. Lightweight binary ciphers are more natural candidates for this purpose, since it is well-known that they lead to better efficiency and security against horizontal attacks than, for example, the AES [BS21]. We selected SKINNY [BJK+16] for two main reasons. First it is a popular TBC, now an ISO standard, with different tweak sizes of 0, n and $2n$ bits (where n is the block length of the cipher) – which directly corresponds to the mid-pSquare variants we put forward. This makes SKINNY a nice comparison target with respect to performance and black-box security of all mid-pSquare variants. Second it has been designed to be efficient on both software and hardware platforms, with side-channel security via masking as an explicit design goal [BJK+16]. Besides, SKINNY has also been instantiated in different variants of Romulus [IKMP20], a finalist in the NIST lightweight cryptography competition. Concrete software performance comparisons between different lightweight Authenticated Encryption with Associated Data (AEAD) primitives, including Romulus, can be found for example in the eBACS (ECRYPT Benchmarking of Cryptographic Systems) database [BL25]. We nevertheless insist that from the side-channel security viewpoint alone, the conclusions we reach for SKINNY would also be observed with other

[1] This example is given for intuition. The LSB is a worst-case leakage function in the prime-field case and security amplification is better with global (e.g., Hamming weight) leakage functions [FMM+24].

binary cipher structures. For example, protecting the implementations of the recently standardized Ascon family (based on a lightweight permutation over a 320-bit state), also requires Boolean masking due to its bit-slice nature [DEMS21]. Such implementations would therefore suffer from the same lack of noise amplification that we see with SKINNY and effectively any design that is masked over $\mathbb{F}_2$. However, from a performance point of view, it is challenging to draw any meaningful conclusions from direct comparisons between tweakable block ciphers and permutation-based primitives like the Ascon family, motivating the selection of SKINNY.

2 Specification of mid-pSquare over $\mathbb{F}^4_{2^{31}-1}$

Here we provide the details of mid-pSquare over $\mathbb{F}^4_{2^{31}-1}$, whose round function (without tweakey addition) is shown in Figure 2. We refer to [GMM+24] for an explanation of the *design rationale* of this scheme. The main differences to the design proposed in [GMM+24] are the parametrization and the tweak schedule – we will explain the design rationale in the following.

2.1 High-Level Structure

The high-level structure of mid-pSquare is given in Figure 1 and directly follows that of the FPM_τ family of tweakable block ciphers introduced in [GMM+24].

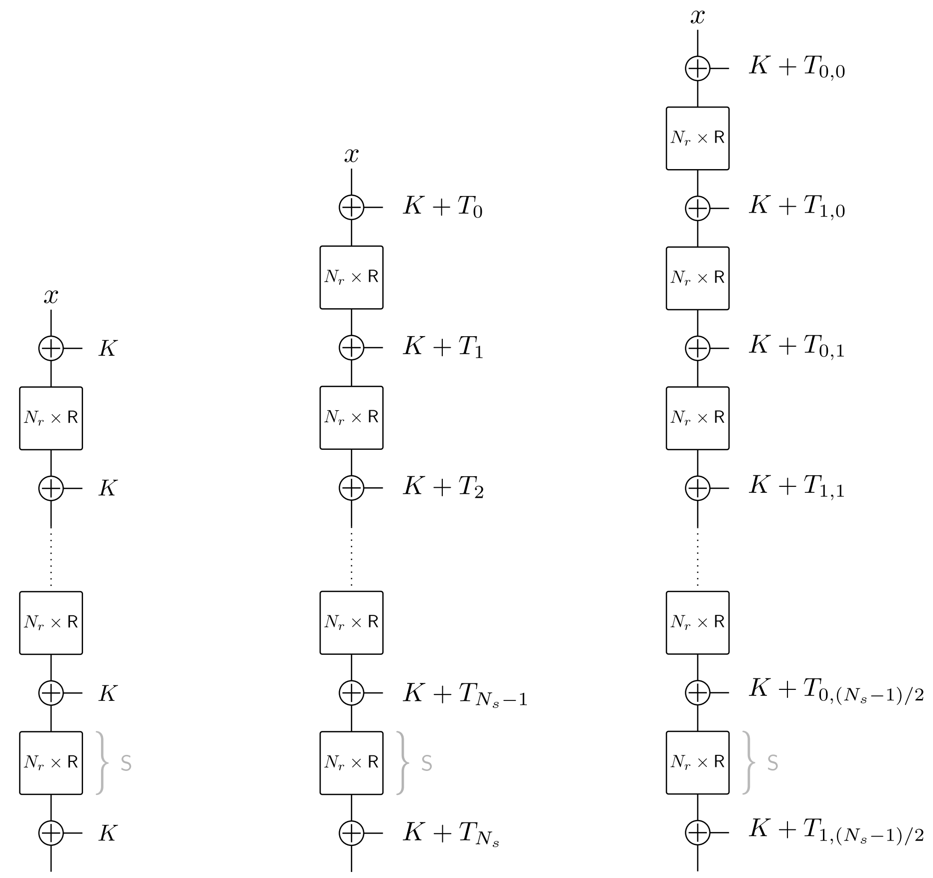


Figure 1: High-level structure of mid-pSquare_τ for $\tau = 0, 1, 2$ (left to right). We use the shortcut notation $N_r \times \mathsf{R}$ to denote the application of the round R N_r times.

The mid-pSquare$_\tau$ tweakable block cipher takes as inputs a plaintext $x \in \mathbb{F}^4_{2^{31}-1}$, a key $K \in \mathbb{F}^4_{2^{31}-1}$ and an optional tweak defined as follows:

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

If $\tau = 0$, mid-pSquare takes no tweak and corresponds to a block cipher. If $\tau > 0$, mid-pSquare is a key alternating cipher, where a tweakey is added every $r > 1$ rounds. We denote a single round as R and a group of N_r rounds R as a step S. A tweakey addition is performed after every step. If $\tau = 0$, the tweakey is always the master key K. If $\tau \geq 1$, the tweakeys are defined as $K + T_{0,0}$, $K + T_{1,0}$, …, $K + T_{\tau-1,0}$, $K + T_{0,1}$, $K + T_{1,1}$, …, $K + T_{\tau-1,1}$, …, $K + T_{0,i}$, $K + T_{1,i}$, …, $K + T_{\tau-1,i}$, where the values $T_{j,i} \in \mathbb{F}_p^n$ are produced by a tweak scheduling algorithm 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 $\mathsf{R} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ (and the steps S) are independent of both the tweak and of the master key.

2.2 Round Function F

The round function F over $\mathbb{F}_p^2$ corresponds to a 2-round Feistel instantiated with $x \mapsto x^2$, where the rotation of the inputs is replaced by a multiplication with a matrix M. It is characterized by two parameters: $b = 2$, the number of branches in the generalized Feistel network and $c = 2$, the number of $\mathbb{F}_p$ values in each branch.

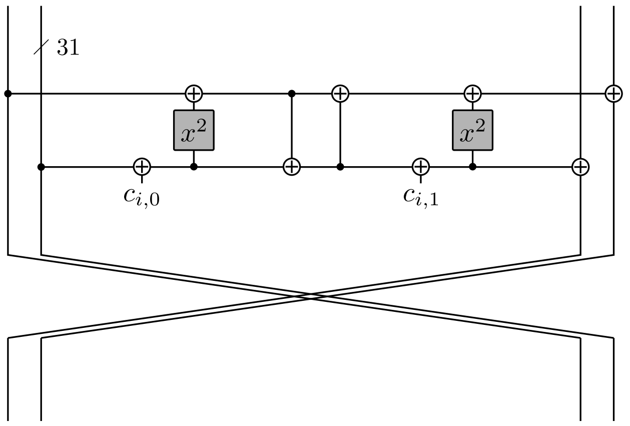


Figure 2: Round R of mid-pSquare over $\mathbb{F}_p^4$ with $b = 2$ and $c = 2$ (see [GMM+24]).

Linear Layer M. The linear layer of mid-pSquare is instantiated via the invertible "pseudo-Hadamard transform" [Mas93] matrix $M \in \mathbb{F}^{2\times 2}_{2^{31}-1}$ defined as:

$$M = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} ,$$

where $x \mapsto 2 \cdot x$ operation is just a bit rotation. The matrix M corresponds to a Maximum Distance Separable (MDS) code, and it can be computed via only two additions with a depth of one if the $x \mapsto 2 \cdot x$ operation is implemented as a bit rotation.

Round Constants. We denote the bit-wise rotation left via $\lll$. Given the first 64 bits of the binary sequence of π – denoted as π_{bin64} – the round constants $c_{i,0}, c_{i,1} \in \mathbb{F}_{2^{31}-1}$ are defined as:

- $c_{i,0} = (\pi_{\text{bin64}} \lll i) \bmod 2^{31}$;
- $c_{i,1} = (\pi_{\text{bin64}} \lll (i+32)) \bmod 2^{31}$.

As no sequence of 31 consecutive 1s exists in π_{bin64}, it holds that $c_{i,0}, c_{i,1} \in \mathbb{F}_{2^{31}-1}$.

2.3 Tweak Schedule ($\tau \geq 1$ Only)

The non-linear tweak schedule performs a sequence of operations reminiscent of the AES/Rijndael round function [DR02]. We first employ a non-linear mapping Ψ to all field elements individually (analogous to `SubBytes`), followed by a row-wise rotation of the state elements Π_τ (analogous to `ShiftRows`), concluded by column-wise matrix multiplication Φ_τ (analogous to `MixColumns`). The resulting function $\Phi_\tau \circ \Pi_\tau \circ \Psi$ realizes a bijective mapping $\mathbb{F}_{2^{31}-1}^{4\cdot\tau} \mapsto \mathbb{F}_{2^{31}-1}^{4\cdot\tau}$, and the tweak update function tku is obtained by iterating it four times. The individual functions Ψ, Π_τ, Φ_τ are defined as follows:

1. Ψ is a non-linear mapping $\mathbb{F}_p \mapsto \mathbb{F}_p$ for $p = 2^{31}-1$ defined as:

$$\Psi(x) = \Lambda^{-1}\left(\Lambda(x) \oplus (\Lambda(x) \gg 1) \oplus \underbrace{(\Lambda(p) \gg 1)}_{\text{constant}}\right),$$

   where $\Lambda : \mathbb{F}_{2^{31}-1} \mapsto \mathbb{F}_2^{31} \setminus \{\mathbf{1}\}$ is the invertible map that returns the bit-string corresponding to the input (where $\mathbf{1} \in \mathbb{F}_2^{31}$ is the 1s bit-string), $\oplus$ is the bitwise XOR, and $\gg$ is a bit-wise right shift. We will prove that this map is bijective below;

2. Π_τ is a linear mapping $\mathbb{F}_{2^{31}-1}^{2\times(2\cdot\tau)} \mapsto \mathbb{F}_{2^{31}-1}^{2\times(2\cdot\tau)}$ defined as:

$$\Pi_\tau \begin{pmatrix} x_0 & x_2 & \dots & x_{4\cdot\tau-2} \\ x_1 & x_3 & \dots & x_{4\cdot\tau-1} \end{pmatrix} = \begin{pmatrix} x_0 & \dots & x_{4\cdot\tau-4} & x_{4\cdot\tau-2} \\ x_3 & \dots & x_{4\cdot\tau-1} & x_1 \end{pmatrix};$$

3. Φ_τ is a linear mapping $\mathbb{F}_{2^{31}-1}^{2\times(2\cdot\tau)} \mapsto \mathbb{F}_{2^{31}-1}^{2\times(2\cdot\tau)}$, in which each column of the state is multiplied by the invertible matrix $M \in \mathbb{F}_{2^{31}-1}^{2\times 2}$ defined as $M = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$.

The resulting function for $\tau = 1$ is depicted in Figure 3.

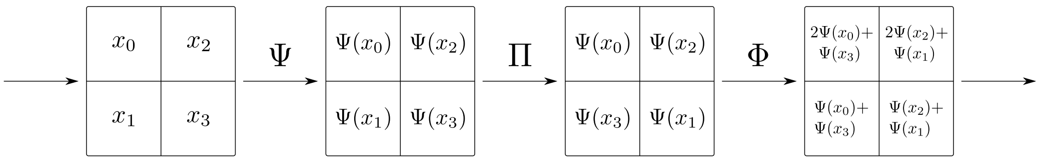


Figure 3: Tweak update function tku for $\tau = 1$ mapping $\mathbb{F}_{2^{31}-1}^4 \mapsto \mathbb{F}_{2^{31}-1}^4$ is realized by iterating the depicted progression *four* times.

Design Rationale. The main related-tweak attacks against small-pSquare are of statistical nature (e.g., differential attacks), as discussed in [GMM+24]. Due to the relatively small size of the prime (approximately 7 bits) and the relatively large state-size (equal to 16), algebraic attacks were indeed less efficient than statistical attacks in that case. We will show in the next section that this is not the case for mid-pSquare: algebraic attacks are now stronger than statistical attacks. In this respect, and since related-tweak algebraic attacks are almost unknown in the literature, the tweak-schedule of mid-pSquare has been designed with the goal to maximize the resistance against such attacks. In order to achieve this goal, mid-pSquare's tweak-schedule aims to have a complex and high degree algebraic representation over $\mathbb{F}_p$ over multiple rounds. In such a way, exploiting the algebraic relation between related tweaks becomes more complicated and expensive from an attacker's point of view. At the same time, it is computed via functions that can be evaluated efficiently over $\mathbb{F}_p$ (as in the case of Π_τ and Φ_τ) or over $\mathbb{F}_2^n$ (as in the case of Ψ). As already pointed out, these functions take inspiration from the AES/Rijndael round function.

Lemma 1. *The tweak schedule* $\mathsf{tku} : \mathbb{F}_{2^{31}-1}^{4\cdot\tau} \mapsto \mathbb{F}_{2^{31}-1}^{4\cdot\tau}$ *is invertible.*

Proof. It is sufficient to prove that Ψ, Π_τ, Φ_τ are invertible. Since Π_τ and Φ_τ are linear mappings we only focus on Ψ. In the case of Ψ, it is sufficient to note that the map $x \mapsto x \oplus (x \gg 1) \oplus 11\ldots10$ over $\mathbb{F}_2^{31}$, that is,

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} \mapsto \begin{bmatrix} 1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 1 & 0 & \ldots & 0 & 0 \\ \vdots & & & & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 & 1 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} \oplus \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \end{bmatrix},$$

is invertible and admits $\mathbf{1}$ as a fixed point. The map over $\mathbb{F}_2^{31}$ is invertible since the matrix has full rank (for each row i, adding all rows with index greater than i to the i-th row gives the canonical vector $\mathbf{e}_i$). The element $\mathbf{1}$ is sent to $00\ldots01 \oplus 11\ldots10 = \mathbf{1}$, that is $\mathbf{1}$ is sent to itself, therefore Ψ is a bijection from $\mathbb{F}_{2^{31}-1}$ to $\mathbb{F}_{2^{31}-1}$. □

We point out that full diffusion in the tweaks is achieved after $2 \cdot \tau$ applications of $\Phi_\tau \circ \Pi_\tau \circ \Psi$, which corresponds to half a tweak update when $\tau = 1$, and a full tweak update when $\tau = 2$. The algebraic properties of this tweak schedule are detailed next.

2.4 Security Level and Number of Rounds

We claim 120 bits of security for mid-pSquare with the aforementioned parameters ($p = 2^{31} - 1; b = 2, c = 2$). In particular, we claim that

> there exists no classical attack in the single-key setting with data, computational and memory complexities smaller than 2^{120}.

For a security level of 120 bits, the number of rounds are defined as follows. Each step S is defined by $N_r = 4$ rounds R. The number N_s of steps S is as follows:

- $\tau = 0$: $N_s = 14$;
- $\tau = 1$: $N_s = 16$;
- $\tau = 2$: $N_s = 18$.

A direct comparison of parameters, sizes and properties to FPM instance small-pSquare is given in Table 1.

Remark 1. An earlier preprint version of this paper[2] included a 124-bit security claim. Yet, after Beyne and Verbauwhede showed that the given degree estimates were overly optimistic [BV25], in particular the minimum number of rounds needed to reach maximum degree (a sufficient but not necessary condition for achieving security), this claim has been revised. While the authors suggested to increase the number of rounds (though admitting "*it is unlikely that there are corresponding attacks*", see [BV25, Sect. 1 – page 4]) we concluded that a small revision of the security claim would be wiser. The degree growth is fast until reaching *almost* maximum degree which is sufficient to support a slightly reduced security claim and only then slows down, requiring approximately the same number of rounds again to eventually reach maximum degree (more details later).

[2] https://eprint.iacr.org/archive/2025/519/20250319:151040

Table 1: Comparison of parameters, sizes and properties of the two FPM instances small-pSquare and mid-pSquare.

FPM Instance	τ	p	b	c	block	secur.	tweak	N_r	N_s	max. deg.	mon.
	0						0		9		
small-pSquare	1	$2^7 - 1$	4	4	112	112	112	4	16	$16 \cdot (p-1)$	p^{16}
	2						224		21		
	0						0		14		
mid-pSquare	1	$2^{31} - 1$	2	2	124	120	124	4	16	$4 \cdot (p-1)$	p^4
	2						248		18		

Hence, instead of increasing the number of rounds we decided to reduce the security claim to 120 bits (which is still above small-pSquare's and should be sufficient for all practical purposes) to restore the initially intended security margin. Our analysis proposed in the following sections supports this choice.

3 Black-Box Security Analysis

In this section, we justify the number of rounds just given for mid-pSquare. Due to the design analogies between mid-pSquare and small-pSquare, the security analysis is similar for the two schemes. For this reason, here we limit ourselves to focus on the main attack vectors against mid-pSquare, by adapting the security analysis of small-pSquare. We refer to the security analysis of small-pSquare presented in [GMM+24] for more details regarding the attack vectors that do not determine the choice of the number of rounds.

To summarize our security analysis, we are going to show that the main attack vectors against mid-pSquare are of algebraic nature, including in particular (related-tweak) interpolation and Gröbner basis attacks. As already pointed out, this results in a crucial difference between the security analysis of mid-pSquare and small-pSquare.

Remark 2. *Related-tweak algebraic attacks* (in which the attacker considers combinations of systems of equations under different chosen tweaks in order to reduce the cost of the attack and/or break more rounds) are almost unknown in the literature. This is presumably due to the fact that symmetric primitives over binary fields such as AES or SHA-3/Keccak have been shown vulnerable to statistical attacks mainly, while algebraic attacks gain more popularity recently due to the emergence of symmetric primitives designed for applications such as Multi-Party Computation (MPC), Fully Homomorphic Encryption (FHE), and Zero-Knowledge (ZK) [GLR+20, GKR+21, DGGK21, BBC+23, GHR+23, GØSW23]. Still, symmetric schemes designed for such applications are not commonly tweakable constructions.

3.1 Statistical Attacks

3.1.1 Differential Attacks

Given pairs of inputs with some fixed input differences, a differential attack [BS91, BS93] exploits the non-uniform probability distribution of the output differences produced by the cryptographic primitive. We recall that the differential probability (DP) of having a certain output difference Δ_O given a particular input difference Δ_I through a permutation P over $\mathbb{F}_p^n$ is defined as:

$$\mathrm{DP}_{\mathsf{P}} = \frac{|\{x \in \mathbb{F}_p^n \mid \mathsf{P}(x+\Delta_I) - \mathsf{P}(x) = \Delta_O\}|}{p^n}.$$

In order to argue for the resistance of mid-pSquare against differential attacks, we compute lower bounds on the minimal number of active square operations, both in the single-key and related-tweakey model. In contrast to the single-key model, where the round tweakeys are constant and thus do not influence the activity pattern, an attacker is allowed to introduce differences within the tweakey state in the related-tweakey model. For achieving this goal, we make use of the following observations:

- Since $x \mapsto x^2$ is a quadratic map, $\text{DP}_{x \mapsto x^2} \leq p^{-1} \approx 2^{-31}$ for each $\Delta_I, \Delta_O \neq 0$;
- The minimum number of active square maps in F is one. Indeed, since F is a 2-round Feistel scheme over $\mathbb{F}_p^2$, the best case for the attacker can occur when the first $x \mapsto x^2$ map is inactive. It directly follows that $\text{DP}_{\mathsf{F}} \leq p^{-1} \approx 2^{-31}$;
- Define the function S as the 4-round Feistel scheme R^4 instantiated with F (and recall that 2 rounds lead to full diffusion). As is well-known (e.g., [DKLS20]), at least one function R is active every 2 rounds. It follows that $\text{DP}_{\mathsf{S}} \leq (\text{DP}_{\mathsf{F}})^2 \leq p^{-2} \approx 2^{-62}$.

We then compute the minimum number of steps N_s required to guarantee security. Due to the clustering effect (i.e., 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 (MitM) approach for setting up the attack, we claim that the scheme is secure if every differential characteristic has probability smaller than $2^{-2.5 \cdot \kappa} \approx 2^{-300}$ for a factor 2.5 (equal to the one used for small-pSquare), where $\kappa = 120 < 31 \cdot 4$ bits 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 to achieve full diffusion. It follows that:

- $\tau = 0$: by simple computation, we have $N_s \geq \lceil 300/62 \rceil + 2 = 5 + 2 = 7$, where 2 steps S are added for preventing partial key-guessing strategies;
- $\tau = 1$: following the argument proposed by LED's designers in [GPPR11] and already used for small-pSquare, 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, $N_s = 2 \cdot 5 + 2 = 12$;
- $\tau = 2$: following the argument proposed by LED's designers in [GPPR11] and already used for small-pSquare, even if the attacker has more freedom in the choice of the tweaks, it is possible to show that at least two among four consecutive steps are active (see e.g. the "super-step" $\mathsf{S}^2 := \mathsf{S} \circ \mathsf{S}$ argument given in [GMM+24, Sect. 5.1]). Similar to before, it follows that $N_s \geq 10$ is a necessary condition for security. Still, the attacker can potentially skip two steps (equivalently, 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 $N_s \geq 2 \cdot 5 + 4 + 2 = 16$.

3.1.2 Other Statistical Attacks

As in the case of small-pSquare, linear attacks [Mat93] do not outperform differential attacks against our scheme. A similar conclusion holds for truncated differential [Knu94], impossible differential [BBS99], boomerang [Wag99] and other statistical attacks. In particular, truncated differential cryptanalysis generalizes differential cryptanalysis in the sense that truncated differentials are differentials where only a part of the difference is specified (e.g., usually the attacker only specifies if the difference is zero or non-zero). Impossible (truncated) differential attacks exploit output (truncated) differences which cannot occur in the course of a function, that is, output (truncated) differences with probability zero. It is well-known that in the case of a Feistel scheme instantiated with a

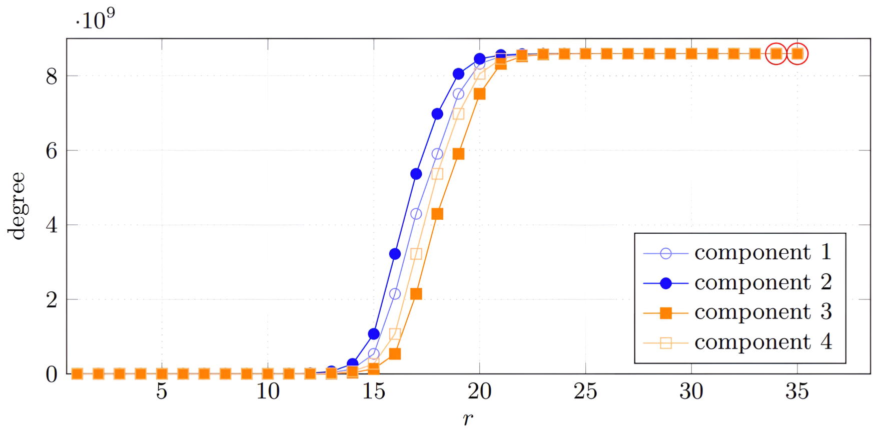


Figure 4: Degree growth of mid-pSquare in terms of the number of rounds r from [BV25, Figure 13].

permutation, impossible differentials exist for 5 rounds [LR88, Pat98, Pat04]. Since each step S consists of 4-round F, the number of rounds required to guarantee security against differential attacks is largely sufficient for preventing these attacks as well.

3.2 Algebraic Attacks

Algebraic attacks exploit the polynomial representation of a cipher for forgery or key recovery. Interestingly, the scenario for mid-pSquare is similar to recently introduced MPC-friendly Pseudo-Random Functions (PRF) `Ciminion` [DGGK21] and Hydra [GØSW23]. We have a relatively large prime field ($p = 2^{31} - 1$), a relatively small state ($n = 4$) and a low degree Feistel round function ($\deg(\mathsf{R}) = 4$). In this setting, algebraic attacks are expected to be more competitive than statistical attacks that are sensitive to the underlying finite field size, which we discuss next.

3.2.1 Degree and Density of the Polynomial Representation

First of all, we evaluate the growth of the degree in mid-pSquare with a fixed key. In the case $\tau = 0$ and/or of a fixed tweak, the polynomials representing mid-pSquare belong to $\mathbb{F}_p[x_0, x_1, x_2, x_3]/(x_0^p - x_0, x_1^p - x_1, x_2^p - x_2, x_3^p - x_3)$ where $p = 2^{31} - 1$. Hence, we note that the degree in one variable is at most $p - 1 = 2^{31} - 2$, that the total degree is then at most $4 \cdot (p - 1)$ and that the total number of monomials is p^4. Before going on, we point out that the following analysis can be generalized for the case of related tweaks, for which the adversary can consider the same polynomials but with more variables.

Growth of the Degree. It can be observed that $\deg(\mathsf{R}) = \deg(\mathsf{R}^{-1}) = 4$, noting that the degree of a Feistel scheme remains the same in both the forward and backward directions. More specifically, at the output of the internal function of R and of R^{-1}, only one of the components attains the maximum degree of 4, while the other has degree 2. Due to the Feistel structure, the degree remains unchanged for half of the state, since this portion does not enter the function F. For this reason, it follows that the minimum degree over the 4 polynomials of a step of r consecutive rounds R (and R^{-1}) is $2^{2 \cdot r - 3}$. More precisely, for $r \geq 2$, the degrees of the four polynomials are $2^{2 \cdot r - 1}, 2^{2 \cdot r}, 2^{2 \cdot r - 3}$ and $2^{2 \cdot r - 2}$. In other words, the minimal degree a polynomial can attain after r rounds is $2^{2 \cdot r - 3}$, until it reaches the maximum degree in one variable. Therefore, the condition $2^{2 \cdot r - 3} \geq p - 1 = 2^{31} - 2$ holds for $r \geq 17$. Then, we consider when the maximum total degree is reached. First,

Table 2: Maximum and minimum degree of mid-pSquare for $r \geq 15$ rounds as predicted in [BV25].

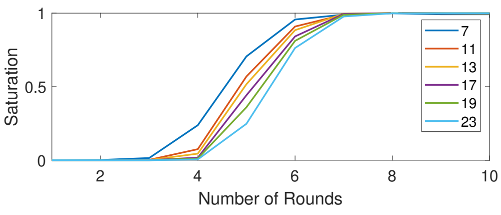
round	$\log_2$(min degree)	$\log_2$(max degree)
15	27.0	30.0
16	29.0	31.584962499825412
17	30.999999999328193	32.32192809354375
18	31.999999998992287	32.70043971690083
19	32.45943161741583	32.90689059417533
20	32.80735492071399	32.977279922134976
21	32.95419630899992	32.99435343550997
22	32.98868468541797	32.9985904284004
23	32.997179479591374	32.999647735184425
24	32.99929538567914	32.999911940851725
25	32.99982387767195	32.9999779847091
26	32.9999559704259	32.99999449520106
27	32.999988991724734	32.99999862279452
28	32.99999724693134	32.999999654691045
29	32.99999931072562	32.999999912665054
30	32.99999982667372	32.99999997715855
31	32.999999955660726	32.99999999328193
32	32.99999998790747	32.99999999731277
33	32.999999995969155	32.99999999832048
34	32.99999999798458	32.999999998488434
35	32.999999998488434	32.999999998488434

we determine the minimal number of rounds such that at least one complete monomial appears, where $x^e = \prod_{i=0}^{3} x_i^{e_i}$ is called complete if $e_i > 0$ for $0 \leq i \leq 3$. We observe that such monomial appears after 2 rounds, and we determine the number of rounds sufficient to obtain monomials up to the maximum total degree in the 4 polynomials: $2^{2\cdot(r-2)-3} \geq 4(p-1) = 4(2^{31}-2)$, which is satisfied for $r \geq 20$ (hence, 3 extra rounds).

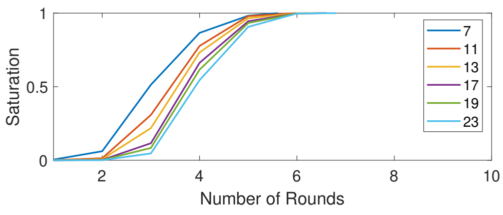
Recent Results by Beyne and Verbauwhede [BV25]. It is crucial to keep in mind that the previous number of rounds is a lower bound, that is, it gives an estimation of the *minimum* number of rounds necessary to reach maximum degree. In practice, due to the particular structure of the cipher, more rounds could be necessary. This is exactly what Beyne and Verbauwhede [BV25] recently proved. As in the case of ciphers over binary fields [BCC11, EGL+20, CGG+22], the growth of degree slows down when it is close to its maximum – a graphical representation for mid-pSquare is given in Figure 4, concrete numbers are shown in Table 2. In particular, Beyne and Verbauwhede showed that 35 rounds are actually necessary for reaching the maximum degree $4 \cdot (p-1) \approx 2^{33}$. At the same time, it is possible to observe that the difference between the maximum degree $4 \cdot (p-1)$ and the degree at round e.g. $r \geq 20$ is very small, and not expected to impact our design with a claimed security level of 120 bits.

Density of the Polynomial Representation. Besides the degree's growth, another crucial factor for preventing algebraic attacks is the density of the polynomial representation. Indeed, with equal degrees, a cipher whose polynomial representation is sparse is more vulnerable with respect to some algebraic attacks than a cipher for which it is full. Before going on, we point out the following.

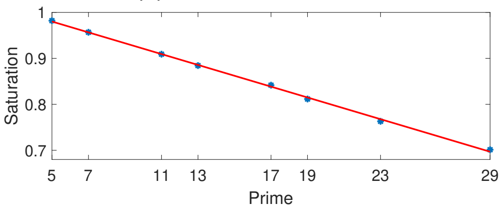
Assumption 1. *We say that the polynomial representation of mid-pSquare has reached*



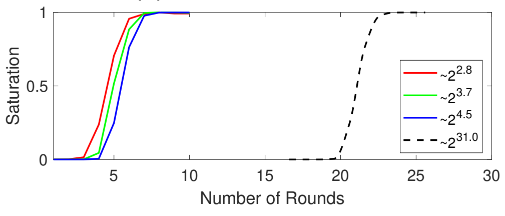
(a) Minimum density.



(b) Maximum density.



(c) Min density after 6 rounds vs. prime



(d) Extrapolation of min density for $p = 2^{31} - 1$.

Figure 5: Minimum and maximum density of the interpolation polynomial of toy mid-pSquare for p in $7 \leq p \leq 23$. Figure 5c shows the (linear) relation between density after 6 rounds and the prime p. In Figure 5d, extrapolation of the number of rounds needed to reach saturation for $p = 2^{31} - 1$.

saturation if it contains $\frac{p-1}{p} \cdot p^4$ *monomials.*[3]

As in the case of small-pSquare, checking the density of the polynomial representation of mid-pSquare is too complex, and out of our reach. For this reason, we focus on a toy-version of mid-pSquare defined over $\mathbb{F}_p^4$ for smaller primes p with $7 \leq p \leq 23$. Our results are displayed in Figure 5. Beyond providing figures for minimum and maximum density we also verified that the density after a certain number of rounds (here 6) seemingly decreases linearly for increasing primes. We then use this relationship to extrapolate the density development for $p = 2^{31} - 1$. Concrete details for $p \in \{19, 23\}$ are given in Table 3. Focusing on the case $p \in \{19, 23\}$, there are respectively 130321 and 279841 monomials in 4 variables, and on average a polynomial over $\mathbb{F}_p^4$ has $\frac{p-1}{p} \cdot p^4$ terms which corresponds respectively to 123462 and 267674 terms. Analogous results hold for $p \in \{7, 11, 13, 17\}$. From our extrapolation (see Figure 5), we estimate that 24 rounds are sufficient for the polynomial to be saturated.[4]

Algebraic Properties of the Tweak-Schedule. For the tweak schedule, the complex representation of $\Psi(x)$ over $\mathbb{F}_p$ prevents us from computing its polynomial directly and checking the system given by the tweak schedule. Hence, we analyze it by studying the properties obtained from the same system over smaller Mersenne primes.

First, we computed the polynomial representation of the univariate function Ψ and its inverse for $p = 7, 31$ and 127. In all cases, the degree is $p - 2$, and the number of monomials indicates high density. Specifically, there are 6 monomials for both Ψ and its inverse when $p = 7$, 27 and 29 monomials for $p = 31$, and 118 and 122 monomials for $p = 127$.

Then, for the polynomial system derived from the tweak schedule, we focused on a toy-version of small-pSquare with $p = 7$ and $\tau = 1$. The polynomial system nearly reaches saturation by the end of the first round (between 1964 and 1984 monomials), and it is already saturated before the end of the second tweak update. As a result, for $\tau = 2$, we

[3] This corresponds to the average for a polynomial in the ring in which we operate.

[4] In [BV25], Beyne and Verbauwhede pointed out that 35 (to reach maximum degree) + 2 (to get density) = 37 rounds are necessary. We emphasize that this refers to a stricter condition on the density than our definition of saturation with $(p - 1) \cdot p^3$ monomials.

Table 3: Number of monomials per branch in the polynomials after r rounds of mid-pSquare over $\mathbb{F}_p$ (with constants fixed to 0). For the considered values of $p \in \{19, 23\}$, there are respectively 130321 and 279841 monomials in 4 variables.

Rnd	Number of monomials
1	4, 9, 1, 1
2	35, 194, 4, 9
3	1443, 10317, 35, 194,
4	44476, 76034 1443, 10317
5	100198, 115181, 44476, 76034
6	122131, 123177, 100198, 115181
7	123492, 123663, 122131, 123177
8	123429, 123471, 123492, 123663

(a) Case: $p = 19$.

Rnd	Number of monomials
1	4, 9, 1, 1
2	35, 196, 4, 9
3	1529, 12402, 35, 196
4	66368, 145824, 1529, 12402
5	204257, 242718, 66368, 145824
6	261575, 267086, 204257, 242718,
7	267397, 267853, 261575, 267086
8	267573, 267826, 267397, 267853

(b) Case: $p = 23$.

expect the polynomial system to reach saturation after 6 iterations of $\Phi_\mathcal{T} \circ \Pi_\mathcal{T} \circ \Psi$, and therefore to also be saturated before the end of the second tweak update.

3.2.2 Linearization Attack

A system of x polynomial equations in x variables can be solved via the linearization technique, which consists in adding new variables to replace all the monomials of degree larger than 1 in the system. The resulting system of equations can be solved using linear algebra if there are enough equations. The computational complexity of this attack is:

$$\min_{0 \le l < x} \mathcal{O}(p^l \cdot \#(D, x - l)^\omega),$$

where l is the number of guessed variables, $\#(D, z) = \binom{D+z}{z}$ is the maximum number of monomials in a polynomial of degree $D < p$ in z variables, and $2 \le \omega < 3$. We remark that MitM versions of this attack are also possible. In the case of mid-pSquare with $\tau = 0$ (remember that the key is composed of 4 $\mathbb{F}_{2^{31}-1}$ words), then:

$$\min_{0 \le l < 4} 2^{31 \cdot l} \cdot \binom{D + 4 - l}{4 - l}^\omega \ge 2^{120}$$

occurs already for $D \ge 2^{16.15}$ (taking the conservative value $\omega = 2$). Since the minimal degree follows 2^{2r-3} (where $2^{2r-3} \ge 2^{16.15}$ for $r \ge 10$) and based on the density analysis just given, we therefore conclude that $2 \cdot 5 = 10$ steps S (equivalently, $2 \cdot 20 = 40$ rounds) are sufficient to prevent MitM algebraic attacks based on linearization.

In the case of mid-pSquare with $\tau \ge 1$, the attacker can potentially simplify the equations to solve by making use of the freedom in the tweak. For example, the linear combinations of several polynomials under properly chosen related tweaks can be exploited to cancel monomials whose coefficients depend on the tweaks or part of them. Still, this procedure is *not* for free, since the attacker has to solve equations in the tweak variables, that are dense and of high (e.g., maximum) degree due to the non-linear tweak schedule. As we have seen, $2 \cdot \tau$ steps are sufficient for reaching full diffusion in the tweak-schedule. At the same time, $2 \cdot \tau$ steps guarantee a complex algebraic representation of the tweak schedule, which is represented by a polynomial of maximum degree and dense. Based on this, we conjecture that $2 \cdot \tau$ extra steps – for a total of $10 + 2 \cdot \tau$ steps, or $40 + 8 \cdot \tau$ rounds – are sufficient to prevent related-tweak algebraic attacks based on linearization.

Relation with Gröbner Bases Attacks. We recall that Gröbner basis attacks reduce to linearization ones when (i) the attacker aims to solve equations linking the plaintexts (and 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).

3.2.3 Interpolation Attack

Interpolation attacks were introduced by Jakobsen and Knudsen [JK97] to break symmetric primitives with a simple (and low degree) algebraic expression. As the name suggests, the attack strategy consists in considering the (intermediate) ciphertext as a polynomial of the plaintext.[5] With sufficiently many plaintext/ciphertext pairs, one can reconstruct this polynomial. Since the polynomial is key-dependent, some (partial) round keys can be recovered by employing a guess-and-determine strategy. As for the linearization attack, we remark that MitM versions of such attack are also possible.

The simplest way to prevent the interpolation attack is to ensure that the polynomial representation is of maximum degree and dense. This implies that the attacker needs (more than) the full-codebook to construct the polynomial, making the attack infeasible.

In the case of mid-pSquare with $\tau = 0$, then 24 rounds (equivalently, 6 steps S) are sufficient to reach close to the maximum degree and density in one direction. Due to the MitM approach, we conjecture that $6 \cdot 2 + 2 = 14$ steps S (equivalently, 56 rounds) are sufficient to prevent algebraic attacks based on interpolation attacks, where 2 steps S are added for preventing partial key-guessing strategies. In the case of mid-pSquare with $\tau \geq 1$, based on the algebraic complexity of both the polynomial systems of the scheme and of the tweak schedule, we add $2 \cdot \tau$ extra steps S (equivalently, $8 \cdot \tau$ rounds) to mitigate related-tweak variants of the interpolation attack, similar to how we handle linearization attacks. This brings the total to $14 + 2 \cdot \tau$ steps S (equivalently, $56 + 8 \cdot \tau$ rounds).

3.2.4 Gröbner Basis Attack

Given at least one known plaintext-ciphertext sample of mid-pSquare (with known tweaks), an adversary can set up a fully determined polynomial system and subsequently solve for the key variables. A general tool for polynomial system solving are *Gröbner bases*, see [CLO13] for an introduction. A Gröbner basis attack proceeds in three steps:

1. Compute a *degrevlex* Gröbner basis.
2. Term order conversion to a *lexicographic* Gröbner basis.
3. Factor the univariate polynomial.

We quickly recall the *lexicographic* and *degrevlex* term orders on a polynomial ring $K[x_1, \dots, x_n]$, where K is a field. Given monomials $a = \prod_{i=1}^{n} x_i^{a_i}$ and $b = \prod_{i=1}^{n} x_i^{b_i}$, then we say that *lexicographically* $a >_{lex} b$ if there exists $1 \leq j \leq n$ such that $a_1 = b_1, \dots, a_{j-1} = b_{j-1}$ and $a_j > b_j$. Moreover, we say that *degree reverse lexicographically* (*degrevlex*) $a >_{DRL} b$ if either $\deg(a) > \deg(b)$, or $\deg(a) = \deg(b)$ and there exists $1 \leq j \leq n$ such that $a_n = b_n, \dots, a_{j+1} = b_{j+1}$ and $a_j < b_j$. For the analysis of mid-pSquare, we consider an iterated polynomial model where variables are introduced for the key and the intermediate states after every round. For ease of presentation, we assume that a key is added after every Feistel layer, i.e. we work with step size $N_r = 1$. We denote the total number of rounds of mid-pSquare by $r = N_r \cdot N_s$. Also, let:

$$y_0 = (x_1 + c_{i,0})^2 + x_0 + x_1 + c_{i,0}, \tag{1}$$

$$y_1 = (y_0 + c_{i,1})^2 + 2 \cdot y_0 - x_1, \tag{2}$$

[5] The variables of such polynomial are the plaintexts (or the ciphertexts) and not the key.

then we abbreviate the i^{th} round of mid-pSquare as:

$$\mathsf{R}_i \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_2 + y_0 + c_{i,1} \\ x_3 + y_1 - c_{i,0} \\ x_0 \\ x_1 \end{pmatrix}. \tag{3}$$

The polynomials modeling the i^{th} round are then given by:

$$\mathbf{f}^{(i)} = \begin{cases} \mathsf{R}_1\left(\mathbf{p} + \mathbf{k} + \mathbf{t}^{(0)}\right) + \mathbf{k} + \mathbf{t}^{(1)} - \mathbf{x}^{(1)}, & i = 1, \\ \mathsf{R}_i\left(\mathbf{x}^{(i-1)}\right) + \mathbf{k} + \mathbf{t}^{(i)} - \mathbf{x}^{(i)}, & 2 \leq i \leq r-1, \\ \mathsf{R}_r\left(\mathbf{x}^{(r-1)}\right) + \mathbf{k} + \mathbf{t}^{(r)} - \mathbf{c}, & i = r, \end{cases} \tag{4}$$

where $\mathbf{k} = (k_0, \dots, k_3)$ denotes the key variables, $\mathbf{x}^{(i)} = \left(x_0^{(i)}, \dots, x_3^{(i)}\right)$ denotes intermediate state variables, $\mathbf{t}^{(i)} = \left(t_0^{(i)}, \dots, t_3^{(i)}\right) \in \mathbb{F}_p^4$ denotes the tweaks, and $\mathbf{p}, \mathbf{c} \in \mathbb{F}_p^4$ is a plaintext-ciphertext pair. Then $\mathcal{F} = \left\{\mathbf{f}^{(i)}\right\}_{1 \leq i \leq r} \subset \mathbb{F}_p\left[\mathbf{x}^{(i)}, \mathbf{k} \mid 1 \leq i \leq r-1\right]$ is a polynomial model for mid-pSquare with a single plaintext-ciphertext sample.

Single Plaintext-Ciphertext Sample. A recent work [Ste25] constructed a quadratic *degrevlex* Gröbner basis for the MPC-friendly PRF Hydra [GØSW23] via affine preprocessing and a linear change of coordinates. By performing a non-linear preprocessing step in every round we can extend this *degrevlex* Gröbner basis construction to mid-pSquare if $2 \mid r$:[6]

1. For $1 \leq i \leq r$, replace $f_1^{(i)} = f_1^{(i)} \mod \left(f_0^{(i)}\right)$ where division with remainder is performed w.r.t. the *degrevlex* term order $\mathbf{x}^{(1)} >_{DRL} \dots >_{DRL} \mathbf{x}^{(r-1)} >_{DRL} \mathbf{k}$.[7] With Equations (1), (2) and (4) we can see that $\deg\left(f_1^{(i)}\right) = 2$ after the reduction.
2. Set up new substitution variables $\hat{x}_1, \dots, \hat{x}_{2\cdot r}$:
   - for $i = 1$, set $\hat{x}_1 = k_1$ and $\hat{x}_2 = x_0^{(1)} - k_0 - k_2$;
   - for $1 < i < r$, set $\hat{x}_{2\cdot i-1} = x_1^{(i-1)}$ and $\hat{x}_{2\cdot i} = x_0^{(i)} - x_2^{(i-1)} - k_0$;
   - for $i = r$, set $\hat{x}_{2\cdot r-1} = x_1^{(r-1)}$ and $\hat{x}_{2\cdot r} = -x_2^{(r-1)} - k_0$.
3. Collect the quadratic polynomials of the $\mathbf{f}^{(i)}$'s in $\mathcal{F}_{squ}$, the linear polynomials of the $\mathbf{f}^{(i)}$'s in $\mathcal{F}_{lin}$, and the polynomial containing the $\hat{x}_i$'s in $\mathcal{F}_{subs}$.
4. Perform Gaussian elimination on $\mathcal{F}_{lin}$ and replace $\mathcal{F}_{squ} = \mathcal{F}_{squ} \mod (\mathcal{F}_{lin})$ and $\mathcal{F}_{subs} = \mathcal{F}_{subs} \mod (\mathcal{F}_{lin})$, where division with remainder is performed w.r.t. the *degrevlex* term order $\mathbf{x}^{(1)} >_{DRL} \dots >_{DRL} \mathbf{x}^{(r-1)} >_{DRL} \mathbf{k} >_{DRL} \hat{x}_1 >_{DRL} \dots >_{DRL} \hat{x}_{2\cdot r}$.
5. Perform Gaussian elimination on $\mathcal{F}_{subs}$ and replace $\mathcal{F}_{squ} = \mathcal{F}_{squ} \mod (\mathcal{F}_{subs})$.

Let us illustrate the idea behind Steps 1 and 2. The polynomials in $\mathbf{f}^{(i)}$, where $1 < i < r$, reduce to:

$$\begin{pmatrix} x_2^{(i-1)} + y_0 + c_{i,1} + k_0 \\ x_3^{(i-1)} + (y_0 + c_{i,1})^2 + 2 \cdot y_0 - x_1 + c_{i,0} + k_1 \\ x_0^{(i-1)} + k_2 \\ x_1^{(i-1)} + k_3 \end{pmatrix} = \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ x_3^{(i)} \end{pmatrix}, \tag{5}$$

[6] We remark that the number of rounds of our scheme is always even.

[7] We naturally order the vectors' variables, i.e., $x_0^{(i)} >_{DRL} \dots >_{DRL} x_3^{(i)}$ and $k_0 >_{DRL} \dots >_{DRL} k_3$.

and similarly, with appropriate modifications for $i = 1, r$. (We omitted the tweaks for ease of presentation.) Then, $y_0 = x_0^{(i)} - x_2^{(i-1)} - k_0 - c_{i,1}$, and substituting this expression into the second branch yields that:

$$\left(x_0^{(i)} - x_2^{(i-1)} - k_0\right)^2 + 2 \cdot \left(x_0^{(i)} - x_2^{(i-1)} - k_0\right) + x_3^{(i-1)} - x_1^{(i-1)} + k_1 + \tilde{c}_i = x_1^{(i)}, \quad (6)$$

where $\tilde{c}_i = c_{i,0} - 2 \cdot c_{i,1} \in \mathbb{F}_p$. This polynomial is exactly the result of $f_1^{(i)} \mod \left(f_0^{(i)}\right)$ with respect to *degrevlex*. Consequently, the quadratic components of $f_0^{(i)}$ and $f_1^{(i)} \mod \left(f_0^{(i)}\right)$ are both squares of linear equations, and we collect these linear equations in the $\hat{x}_i$'s. The remaining steps of the procedure essentially perform a change of variables to the $\hat{x}_i$'s.

If $\mathcal{F}_{lin}$ and $\mathcal{F}_{subs}$ both have full rank, then the change of variables to the $\hat{x}_i$'s is fully determined and invertible. In particular, we then have that $\mathcal{F}_{squ} \subset P = \mathbb{F}_p\left[\hat{x}_j \mid 1 \leq j \leq 2 \cdot r\right]$ and for every $1 \leq j \leq 2 \cdot r$ there exists $f_j \in \mathcal{F}_{squ}$ such that $\mathrm{LM}_{DRL}(f) = \hat{x}_i^2$. By [CLO13, Chapter 2 §9 Theorem 3, Proposition 4] $\mathcal{F}_{squ}$ is then a *degrevlex* Gröbner basis for mid-pSquare. In addition, it is easy to see that the $\mathbb{F}_p$-vector space dimension of $P/(\mathcal{F}_{squ})$ is given by $\dim_{\mathbb{F}_p}(\mathcal{F}_{squ}) = 2^{2 \cdot r}$. In practice, for our model with $N_r = 1$, we observed that this substitution process is always successful over prime fields $\mathbb{F}_p$ and for any even round number r.[8] The construction of the Gröbner basis can be extended to the full mid-pSquare with $N_r = 4$, but the $\hat{x}_i$'s have to be slightly modified, because the key is only added after every full step. Then, we observed that the substitution process is successful for any $N_s \geq 1$. Therefore, in a Gröbner basis attack on mid-pSquare we can proceed directly to the term order conversion step. Using the state-of-the-art FGLM (Faugère, Gianni, Lazard and Mora) algorithm [FGHR14] this step has a polylogarithmic complexity of $\tilde{\mathcal{O}}\left(2^{\omega \cdot 2 \cdot r}\right)$, where $2 \leq \omega < 3$ is the linear algebra constant [VXXZ24]. Thus, to achieve security level κ with respect to term order conversion, we must ensure that $r \geq \frac{\kappa}{2 \cdot \omega}$, hence conservatively $r \geq \frac{\kappa}{4}$.

More Plaintext-Ciphertext Pairs (with No/Single/Un-Releated Tweaks). Assume we are given $k \geq 2$ plaintext-ciphertext samples, but k is too small to perform a linearization attack (see Section 3.2.2). In this case, we can still set up an overdetermined iterated polynomial system for mid-pSquare consisting of $k \cdot 4 \cdot r$ equations in $k \cdot 4 \cdot (r-1) + 4$ variables. Analogous to the construction of the *degrevlex* Gröbner basis, we can reduce this system to $k \cdot 2 \cdot r$ quadratic equations in $k \cdot 2 \cdot r - (k-1) \cdot 4$ variables. Each sample then contains two polynomials with the quadratic terms:

- k_1^2, which comes from the first round, and
- k_3^2, which comes from the last round.

The term k_1^2 is obvious from the first round function. For the term k_3^2, recall that one of the quadratic equations in the last round is $x_2^{(r-1)} + \left(x_1^{(r-1)} + c_{r,0}\right)^2 + c_{r,1} + k_0 = c_0$, but we also have the linear equation $x_1^{(r-1)} + k_3 = c_3$. Hence, we produce the claimed quadratic term after a linear variable elimination. So, we can produce additional $(k-1) \cdot 2$ linear polynomials to eliminate the same number of variables. After this preprocessing, we are left with $k \cdot 2 \cdot r - (k-1) \cdot 2$ quadratic polynomials in $k \cdot 2 \cdot r - (k-1) \cdot 6$ variables.

For the complexity estimation, we assume that this quadratic system is semi-regular (see, e.g., [BFSY05] for a formal definition). The degree of regularity D_{reg} is defined next, as the index of the first non-positive coefficient in the series:

$$\mathsf{H}(z) = \frac{\left(1 - z^2\right)^{k \cdot 2 \cdot r - (k-1) \cdot 2}}{(1 - z)^{k \cdot 2 \cdot r - (k-1) \cdot 6}}. \quad (7)$$

[8] This process always fails for odd round numbers, but we use an even number of rounds in our scheme.

The complexity of computing a DRL Gröbner basis with the F5 algorithm [Fau02] is then bounded by (see [BFSY05]):

$$\mathcal{O}\left(\binom{k \cdot 2 \cdot r - (k-1) \cdot 6 + D_{\text{reg}}}{D_{\text{reg}}}^{\omega}\right), \tag{8}$$

where $2 \leq \omega \leq 3$ is the linear algebra constant.[9] For example, for $r = 16$ (equivalently, 4 steps) and $k = 2$ we have $D_{\text{reg}} = 24$, and with $\omega = 2$ the binomial coefficient evaluates to approximately 136 bits. For $r = 12$ (equivalently, 3 steps) and $k = 3$, we have $D_{\text{reg}} = 21$ and with $\omega = 2$ the binomial coefficient evaluates to approximately 127 bits.

Since this approach performs worse than term order conversion even for $k = 2$ we conjecture that recomputing the *degrevlex* Gröbner basis for an overdetermined mid-pSquare system is not more competitive than direct term order conversion.

We finally note that the adversary can linearize more quadratic polynomials in the first round via chosen plaintexts or related tweaks. We discuss this improvement in Appendix C, but we stress that the expected complexity improvement is negligible.

Setting the Number of Rounds. To explain our rationale for choosing the round number of mid-pSquare with respect to Gröbner basis attacks, we need to delve into more details of term order conversion algorithms. Let $\mathcal{G} = \{g_i\}_{1 \leq i \leq 2 \cdot r} \subset P = \mathbb{F}_p\left[\hat{x}_i \mid 1 \leq i \leq 2 \cdot r\right]$ represent our mid-pSquare *degrevlex* Gröbner basis with $\text{LM}_{DRL}(g_i) = \hat{x}_i^2$. Let $\mathcal{B} \subset P$ consist of all square-free monomials, including 1. Then, $\mathcal{B}$ forms an $\mathbb{F}_p$-vector space basis of the quotient ring $P/(\mathcal{G})$ and $|\mathcal{B}| = 2^{2 \cdot r}$. Now, we set up the so-called multiplication matrix $\mathbf{M}_{2 \cdot r}$ for the last variable $\hat{x}_{2 \cdot r}$. The columns of $\mathbf{M}_{2 \cdot r}$ are labeled by $\mathcal{B}$ and the rows by $\hat{x}_{2 \cdot r} \cdot \mathcal{B}$. The entries of row $\hat{x}_{2 \cdot r} \cdot s$ are the coefficients of the remainder $r_{\hat{x}_{2 \cdot r} \cdot s} = \hat{x}_{2 \cdot r} \cdot s \mod (\mathcal{G})$.

Under a standard assumption,[10] the minimal polynomial of $\mathbf{M}_{2 \cdot r}$ corresponds to the univariate polynomial in the *lexicographic* Gröbner basis. Improvements in term order conversion algorithms generally arise from fast linear algebra [FGHR14] or the exploitation of structural properties like *sparsity* [FM17] to compute the minimal polynomial.

We aim to protect mid-pSquare against sparse FGLM algorithms which could exploit additional structural information currently unknown to us. For this purpose, we recorded the sparsity ratio $\#(\text{non-zero entries})/2^{4 \cdot r}$ of the multiplication matrix $\mathbf{M}_{2 \cdot r}$. Since the matrix size grows with $2^{4 \cdot r}$, it is not feasible to collect data for mid-pSquare with step size $N_r = 4$ and $r = N_r \cdot N_s$. Instead, we use the model with $N_r = 1$ as estimator for the sparsity of mid-pSquare. For $r \in \{2, 4, 6\}$, we recorded 100 samples with random keys, constants and plaintexts, and computed the sparsity ratio. We observed that the sparsity ratio follows an exponential decay of the form $\delta(r) = a \cdot 2^{-b \cdot r}$. Our interpolated coefficients are:

$$\log_2\left(\delta(r)\right) = \left(-1.05301 \pm 3 \cdot 10^{-5}\right) \cdot r + \left(0.0597 \pm 1 \cdot 10^{-4}\right), \tag{9}$$

where the $\pm$ terms represents the R^2-value of the respective coefficients. Therefore, in logarithm in base 2 we approximate the number of non-zero coefficients in $\mathbf{M}_{2 \cdot r}$ with:

$$4 \cdot r + \log_2\left(\delta(r)\right) \approx 4 \cdot r - 1.05 \cdot r + 0.06 \approx 2.95 \cdot r. \tag{10}$$

To protect mid-pSquare against sparse FGLM algorithms and achieve a security level of κ, we require that $r \geq \frac{\kappa}{2.95}$. Our rationale is that unless the characteristic polynomial of

[9] We are aware that the bound just given is only an upper bound. At the same time, we emphasize that (1st) no non-trivial lower bound is known in the literature (to the best of our knowledge), and (2nd) this bound has been widely used in the design of several arithmetization-friendly symmetric primitives such as, for example, [GLR+20, GKR+21, DGGK21, BBC+23, GHR+23, GØSW23].

[10] Namely, we assume that the *lexicographic* Gröbner basis must be in $\hat{x}_{2 \cdot r}$-shape position, meaning that it takes the form $\left(\hat{x}_1 - f_1\left(\hat{x}_{2 \cdot r}\right), \dots, \hat{x}_{2 \cdot r - 1} - f_{2 \cdot r - 1}\left(\hat{x}_{2 \cdot r}\right), f_{2 \cdot r}\left(\hat{x}_{2 \cdot r}\right)\right)$.

$\mathbf{M}_{2\cdot r}$ can be trivially computed, a term order conversion algorithm should process every non-zero matrix entry at least once. Consequently, it would require at least $\approx 2^{2.95\cdot r}$ non-trivial operations. For $r = 42$ (meaning 10.5 steps) we have $2.95 \cdot r = 123.9$, and for $r = 48$ (meaning 12 steps) we have $2.95 \cdot r = 141.6$. We conjecture that this worst-case scenario approach provides a sufficient security margin for mid-pSquare with $N_r = 4$ and $N_s = 16$, i.e., $r = 56$ full rounds, against Gröbner basis attacks, even under (hypothetical) algorithmic improvements.

3.2.5 Higher-Order Differential Attacks

Higher-order differential attacks have been introduced by Lai [Lai94] and by Knudsen [Knu94] to attack ciphers over $\mathbb{F}_2^n$ with low degree. In particular, they proved that, given a function F over $\mathbb{F}_2^n$ of degree δ, and given any affine subspace $\mathcal{V} \subseteq \mathbb{F}_2^n$ of dimension $\delta + 1$ (hence, $\dim(\mathcal{V}) \geq \deg(\mathsf{F}) + 1$), then

$$\sum_{x\in\mathcal{V}} x = \sum_{x\in\mathcal{V}} \mathsf{F}(x) = 0\,.$$

Since this fact holds independently of the value of the secret key, it can be used to distinguish a cipher from a pseudo-random permutation, and to set up key-recovery attacks.

An analogous property holds for ciphers over $\mathbb{F}_p^n$, as shown in [BCD+20]. Given a function F over $\mathbb{F}_p^n$ of degree *strictly* less than $d \cdot (p - 1)$, and given any affine subspace $\mathcal{V} \subseteq \mathbb{F}_p^n$ of dimension d, then

$$\sum_{x\in\mathcal{V}} x = \sum_{x\in\mathcal{V}} \mathsf{F}(x) = 0\,.$$

More recently, Beyne and Verbauwhede [BV25] extended such a result by introducing "ultrametric integral cryptanalysis" over finite rings of prime characteristic p.

As we saw before, 35 rounds are necessary for mid-pSquare in order to reach maximum degree. (Note that the data complexity of the attack is at most 2^{120}, which implies that it is unlikely for the attacker to cover such a number of rounds with a zero-sum distinguisher.) Regarding the key-recovery phase, two approaches can be considered. Let $\{p_i\}_i$ be a set of plaintexts for which the zero-sum holds after a certain number of rounds, and let $\{c_i\}_i$ be the corresponding ciphertexts. As first approach, the attacker partially guesses the key, partially decrypts the ciphertexts with respect to the guessed key, and checks that the zero-sum holds. Due to the wrong-key randomization hypothesis, the attacker can discard wrongly guessed keys (that is, the keys for which the zero-sum property does not hold). However, this attack strategy does not allow to cover many rounds, since a single step S provides full diffusion, and the attacker is forced to guess the entire key in order to partially decrypt the ciphertexts for more rounds. As in [EGL+20], another more efficient approach consists in setting up a system of equations defined as

$$\sum_i \mathsf{R}_k^{-r}(c_i) = 0\,,$$

and solve it with respect to the key k (for a certain number of rounds r), instead of partially guessing the key directly. Our previous analysis shows that the attacker cannot solve such a system of equation when $r \geq 20$, which implies that $35 + 20 = 55$ rounds (equivalently, 14 steps) are more than sufficient to prevent higher-order differential attacks for $\tau = 0$.

Similar to how we handle previous algebraic attacks, for $\tau \in \{1, 2\}$, we add $2 \cdot \tau$ extra steps S (equivalently, $8 \cdot \tau$ rounds) to mitigate related-tweak variants of the higher-order differential attack, for a total to $14 + 2 \cdot \tau$ steps S (equivalently, $56 + 8 \cdot \tau$ rounds R).

4 Performance Comparison

Now that we have specified the design of mid-pSquare and analyzed its security against cryptanalysis, it is time to establish its performance when implemented on the envisioned target platform, namely simple 32-bit microcontrollers such as ARM Cortex-M devices.

Choice of the Prime. Both AES-prime [MMMS23] and the original FPM instance called small-pSquare [GMM+24] make use of a 7-bit prime modulus. In particular, they use $p = 2^7 - 1 = 127$, a so-called Mersenne prime known to enable efficient field arithmetic [MMMS23]. The choice of the small field size is intended to maintain efficiency in hardware implementation. However, on 32-bit microcontrollers, optimized instructions for integer arithmetic like addition, subtraction and – most importantly – multiplication of 32-bit operands are commonly available. In most cases, the number of cycles these instructions take to execute is independent of the size of the operand (i.e., they run in constant time). While addition and subtraction require only a single cycle on virtually all devices we consider, multiplication of full-width operands may take multiple cycles on lowest-end processors, such as ARM Cortex-M0/M3. However, in ARM Cortex-M4 devices and almost all more advanced families of 32-bit microcontrollers, unsigned integer multiplication of 32-bit words is also executed in only one cycle for any operand values. Thus, to optimally leverage the capabilities of such Arithmetic Logic Units (ALUs), Mersenne primes of a size close to the maximum single-precision operand width (i.e., 32 bits) should be chosen. Fortunately, $p = 2^{31} - 1$ is a Mersenne prime and appears to be perfectly suited for efficient field arithmetic under the given constraints, as elements from the associated prime field $\mathbb{F}_p$ occupy almost the entire register width, while – very importantly – still allowing to perform one arithmetic addition of two field elements without risking an overflow. This enables optimal efficiency for constant-time modular addition, subtraction and multiplication using the common Mersenne prime reduction trick (see [MMMS23, GMM+24]). Smaller Mersenne primes (e.g., $2^7 - 1$, $2^{13} - 1$, $2^{17} - 1$ or $2^{19} - 1$) would lead to the same number of cycles for individual field operations despite processing fewer bits per instruction.[11] Larger Mersenne primes ($\geq 2^{61} - 1$) would imply costly multi-precision arithmetic. Non-Mersenne primes of equivalent size would require a larger number of cycles per modulo reduction. We therefore argue that mid-pSquare with $p = 2^{31} - 1$ is ideally suited for embedded devices with 32-bit operand/register width, especially if single-cycle full-width multiplication is available, as it enables highly efficient field arithmetic, including the non-linear layers, in constant time.

Choice of the Competitors. A natural target for comparison is of course small-pSquare as the only other existing FPM instance to date. Yet, its hardware-oriented design choices, in particular its small prime modulus and the bit/word shuffling in the tweak schedule, make it less suited for 32-bit software platforms. Therefore, it is significantly more meaningful to select an additional competitor that is actually meant to be efficient in the envisioned setting. For this purpose we have chosen SKINNY, a tweakable lightweight block cipher introduced at CRYPTO 2016 [BJK+16]. Due to its robust security and high efficiency in both software and hardware, SKINNY has been selected as part of the ISO/IEC 18033-7:2022 standard for TBCs. In the following we compare our masked and unmasked implementations of small-pSquare and mid-pSquare to corresponding SKINNY implementations. We believe SKINNY is a great fit for this purpose, as it offers multiple versions with different tweak sizes (namely, 0, n and $2n$, just as we do for mid-pSquare), it has been designed to be efficient on microcontrollers, and it has been designed with masking as a side-channel countermeasure

[11] Depending on the concrete prime choice, underlying algorithm and processor operand width it may be feasible to achieve significant performance improvements by reducing the frequency of modulo reductions when smaller primes are used. Yet we found this to be less effective for our purposes than increasing the prime size.

in mind. In particular, it was demonstrated in the original proposal that bit-sliced SKINNY implementations lead to high performance in software [BJK+16]. At CHES 2021, an even better form of sliced representation of the cipher has been introduced which is called *fix*-sliced representation and allows to minimize the impact of the ShiftRows operation which often becomes a performance bottleneck in bit-sliced software [AP21]. For the following comparison we have taken the efficient fix-sliced implementations provided by the authors of [AP21] and masked them to obtain high performance side-channel resistant software implementations of SKINNY-128-128, SKINNY-128-256 and SKINNY-128-384. We compare the execution time and code size to our respective small-pSquare and mid-pSquare implementations for $\tau = 0$, 1 and 2, up to 32 shares. All masked software implementations are based on the trivial composability notion Probe Isolating Non-Interference (PINI) [CS20] using proven gadgets from [CGLS21] and [CMM+23]. All implementations are constant-time C-language implementations and publicly available under an open-source license at https://github.com/uclcrypto/mid-pSquare.

Results. Tables 4 and 5 show the respective performance figures (in terms of cycle counts) and cost figures (in terms of code size), respectively, for different tweak sizes and numbers of shares d in the masked implementations. All numbers are obtained using Keil MDK-ARM Tool Professional Version 5.36.0.0 using the µVision IDE. The C code is compiled using the default ARM compiler v6 of the tool, with the -O3 flag to optimize for speed, as well as Link Time Optimization (LTO) enabled. Cycle counts are measured on the STM32F415RGT6 microcontroller which is part of the STM32F target board for the ChipWhisperer CW308 UFO board, via ST-LINK in-circuit debugger. They are given for encryption and decryption separately and include everything needed, the data path, the key schedule, the tweak schedule (which remains unmasked for both ciphers due to tweaks being considered public information) and the transformation into the target representation (for fix-sliced SKINNY). This differs from the numbers in [AP21], which exclude some of these parts. The program code size is given for encryption and decryption together as the actual size in bytes of the machine code generated by the compiler and stored in flash memory.

Randomness Generation. For the masked implementations we distinguish between TRNG on and TRNG off. For the TRNG on case we have employed the integrated on-board True Random Number Generator (TRNG) of the Cortex-M4 for masking randomness generation in chunks of 32 bits. The corresponding source code includes TRNG initialization (TRNG_Init) and de-initialization (TRNG_DeInit) functions which are called once each, as well as a randomness extraction function (TRNG_GetRand) to obtain a fresh chunk of 32 random bits whenever needed (this can be many times depending on the masking order and round number). We have also experimented with Pseudo Random Number Generators (PRNGs), but found that, on this target, the on-chip TRNG is more cycle-efficient than deterministic primitives with even a minimal level of cryptographic strength. For the TRNG off case the source code is simply missing the initialization, de-initialization and randomness extraction functions and the constant RAND_CONST is used whenever a random 32-bit value would be needed. Obviously, this leads to an insecure implementation, but we believe it is crucial to report performance numbers both including and excluding the additional cost of randomness generation, as this aspect can be optimized independently of the masked implementations. The differences in the tables between the TRNG on and TRNG off case in terms of both cycles count and code size can then largely be attributed to the presence or absence of calls to the randomness extraction function, respectively.

Interpretation. small-pSquare is clearly the least performing algorithm across the board in terms of cycle counts. Compared to the corresponding mid-pSquare implementations

Table 4: Cortex-M4 cycle counts for SKINNY-128, small-pSquare and mid-pSquare implementations for different tweak sizes and numbers of shares d used when masked.

Primitive		$d = 1$	$d = 2$		$d = 4$		$d = 8$		$d = 16$		$d = 32$	
			TRNG off	on	TRNG off	on	TRNG off	on	TRNG off	on	TRNG off	on
SKINNY-128-128	E	6.9k	17.7k	20.0k	62.9k	77.9k	186.2k	244.6k	830.8k	1.11M	2.56M	3.86M
	D	6.9k	17.8k	20.4k	63.5k	78.5k	188.0k	246.4k	790.2k	1.07M	2.56M	3.86M
small-pSquare $\tau = 0$	E	10.3k	35.1k	56.4k	110.0k	227.0k	557.2k	1.01M	1.91M	3.92M	7.03M	15.4M
	D	10.8k	35.9k	57.4k	108.8k	227.5k	557.0k	1.00M	1.95M	3.92M	7.05M	15.4M
mid-pSquare $\tau = 0$	E	2.9k	11.0k	14.1k	30.8k	55.4k	165.9k	263.2k	574.4k	990.0k	2.08M	3.84M
	D	2.9k	10.8k	14.0k	31.4k	55.3k	163.7k	258.3k	579.5k	992.9k	2.16M	3.82M
SKINNY-128-256	E	8.4k	21.3k	24.1k	75.8k	93.8k	223.6k	293.6k	996.8k	1.33M	3.06M	4.63M
	D	8.4k	21.4k	24.6k	76.4k	94.4k	225.7k	295.8k	954.5k	1.29M	3.06M	4.63M
small-pSquare $\tau = 1$	E	23.6k	67.6k	102.5k	197.3k	405.9k	988.9k	1.80M	3.43M	6.97M	12.7M	27.5M
	D	24.8k	69.1k	106.0k	199.7k	409.5k	990.1k	1.80M	3.45M	6.95M	12.5M	27.5M
mid-pSquare $\tau = 1$	E	5.1k	14.5k	17.7k	37.7k	63.2k	189.5k	300.8k	653.9k	1.14M	2.38M	4.36M
	D	5.1k	13.9k	18.2k	37.6k	65.1k	187.1k	299.3k	668.4k	1.13M	2.46M	4.36M
SKINNY-128-384	E	10.4k	25.4k	28.7k	89.1k	110.2k	261.4k	343.1k	1.16M	1.55M	3.57M	5.41M
	D	10.3k	25.5k	29.2k	89.8k	110.9k	263.9k	345.6k	1.10M	1.49M	3.57M	5.41M
small-pSquare $\tau = 2$	E	30.7k	88.6k	135.2k	258.4k	532.2k	1.30M	2.36M	4.50M	9.15M	16.6M	36.1M
	D	32.1k	90.2k	138.6k	261.5k	537.0k	1.30M	2.34M	4.53M	9.12M	16.4M	36.1M
mid-pSquare $\tau = 2$	E	7.4k	17.5k	22.4k	44.5k	73.3k	215.6k	340.4k	753.3k	1.28M	2.77M	4.92M
	D	7.3k	17.6k	22.2k	44.6k	72.8k	216.4k	337.2k	744.2k	1.28M	2.67M	4.92M

Table 5: Cortex-M4 code size in bytes for SKINNY-128, small-pSquare and mid-pSquare implementations for different tweak sizes and numbers of shares d used when masked.

Primitive	$d = 1$	$d = 2$		$d = 4$		$d = 8$		$d = 16$		$d = 32$	
		TRNG off	on	TRNG off	on	TRNG off	on	TRNG off	on	TRNG off	on
SKINNY-128-128	23.9k	57.5k	66.6k	109.2k	109.6k	243.1k	244.6k	617.1k	618.8k	574.3k	574.4k
small-pSquare $\tau = 0$	3.7k	12.6k	15.9k	21.3k	39.7k	44.8k	36.7k	55.1k	47.7k	76.7k	68.4k
mid-pSquare $\tau = 0$	2.1k	3.6k	5.0k	6.8k	9.8k	13.5k	16.4k	16.6k	18.4k	19.5k	21.2k
SKINNY-128-256	28.1k	68.2k	79.1k	131.7k	132.1k	243.8k	243.5k	756.2k	757.8k	688.0k	688.0k
small-pSquare $\tau = 1$	6.1k	14.9k	18.8k	25.6k	43.9k	49.0k	41.0k	59.6k	52.0k	81.0k	72.7k
mid-pSquare $\tau = 1$	3.0k	4.4k	5.9k	7.8k	10.5k	14.4k	17.3k	17.9k	19.6k	20.5k	22.2k
SKINNY-128-384	32.4k	79.0k	91.6k	154.8k	155.2k	340.8k	342.3k	857.6k	859.2k	801.9k	801.9k
small-pSquare $\tau = 2$	6.3k	15.2k	19.0k	25.8k	44.1k	49.2k	41.2k	59.9k	52.2k	81.3k	72.7k
mid-pSquare $\tau = 2$	5.1k	6.5k	8.2k	12.4k	16.4k	23.3k	27.7k	28.1k	28.8k	34.1k	35.2k

it is between 3× and 7× slower for the same parametrization and masking order. This highlights the difference that dedicated software-oriented and hardware-oriented design choices can make for FPM instances. The gap between SKINNY and mid-pSquare is much smaller. Interestingly, the mid-pSquare software implementations almost consistently (e.g., for any order and tweak size in the TRNG off case) outperform the efficient (masked) fix-sliced SKINNY implementations in terms of cycle counts. In contrast to previous prime-field cipher designs such as AES-prime and small-pSquare, mid-pSquare even outperforms its competitor in the unprotected case. The SKINNY implementations come at a significant code size which is in part caused by the fix-sliced representation, whereas the masked prime-field ciphers can be more than an order of magnitude smaller in memory. In general, the code of mid-pSquare is extremely simple and compact, leading to short compilation times and small binaries even at high masking orders. We conclude that mid-pSquare is highly competitive on 32-bit platforms compared to existing prime-field ciphers and even outperforms certain standardized binary lightweight TBCs, both with and without masking applied.

5 Physical Security Evaluation

In this section, we aim to compare the side-channel security of masked mid-pSquare implementations with the ones of SKINNY, on the same Cortex-M4 platform as we used for performance evaluations. Given the acknowledged difficulty of evaluating side-channel security [ABB+20], especially for ciphers with significantly different structures, we start by providing some rationale for the following experimental analyzes.

In general, side-channel security evaluations can be performed in two fashions. On the one hand, (qualitative) leakage detection aims to provide easy-to-operate conformance-style tests [SM16]. However, if used as a stand-alone tool, it provides limited security guarantees [WO19]. Hence, leakage detection is frequently complemented with a quantitative step, where evaluators try to assess the security of an implementation against the best attacks [SMY09]. In the context of this paper, we are interested in this second (quantitative) approach, both because the horizontal attacks we aim to prevent with prime-field masking are in this category [BCPZ16, BS21], and because we want to compare the side-channel security of binary and prime ciphers with their corresponding masking scheme.

Comparing the side-channel security of different ciphers and masking schemes nevertheless remains non-trivial. Besides, the choice of a 31-bit prime modulus for mid-pSquare implies that predicting its intermediate computations (as needed to perform DPA) requires profiling 31-bit operations for the type of horizontal attacks we are interested in and guessing 31-bit key chunks. Both are computationally-intensive tasks [YK21], and when applied in the context of higher-order side-channel attacks, which use many leakage measurements and require combining the leakage of several shares, they are hardly achievable with current tools [CDSU23]. As a result, our following experiments exploit a number of simplifications and extrapolations. We next detail them for both SKINNY and mid-pSquare and explain why they allow putting forward relevant conclusions and scaling trends.

Starting with SKINNY, we observe that as for any binary cipher implementation, it comes with the benefit that adversaries can trade side-channel signal for (guessing) computational power. That is, by guessing more key bits, one can predict (and exploit the leakage of) more bits of more target intermediate values. Due to the aforementioned computational limitations, we for now limited our investigations to 16-bit key guesses. So a more powerful side-channel adversary (performing 32-bit key guesses) would be able to further decrease the data complexity of the attacks we exhibit. Given this choice, we then tried two guessing strategies. One favors the exploitation of the key addition and MixColumns operations (by partitioning key guesses in order to predict their intermediate values). The other favors the exploitation of the S-box layer. We tested both and observed that the second approach led to the best attacks. So we next report the results obtained with this strategy only.[12]

Following with mid-pSquare, and as just mentioned, we cannot leverage the same side-channel signal vs. computational power tradeoff, since predicting the outputs of the first-round key additions and square operations directly requires guessing 31-bit chunks. Since this is too expensive given the number of traces required to evaluate our masked mid-pSquare implementations, we opted for a different strategy than for SKINNY and evaluated the security of size-reduced toy versions (the entire structure remains the same only the prime field is decreased in cardinality): one with a 7-bit modulus as for small-pSquare [GMM+24], one with a 13-bit modulus and one with a 19-bit modulus (all Mersenne primes, with consistent gaps of 6 bit in between and 2×6-bit to the actual 31-bit prime). Given the theoretical finding that increasing the field size amplifies security without any noise for the Hamming weight leakage function [FMM+24], our goal is to confirm this security guarantee for the concrete leakage function of our target device.

[12] It is an interesting problem to try combining the two strategies. This could however only weaken the security reported for SKINNY, and strengthen our conclusions on the interest of prime masking.

This would imply that the side-channel security of masked mid-pSquare implementations is improved when increasing the modulus, even against adversaries able to guess $|\mathbb{F}_p|$ key candidates. Therefore, it would also imply that any security gap that we observe between our SKINNY and toy mid-pSquare implementations could only be further amplified if considering more powerful adversaries able to directly guess 31/32-bit key chunks.

Setup. Measurements are performed on an ARM Cortex-M4 microcontroller mounted on a ChipWhisperer CW308 UFO board operating at 8 MHz clock frequency derived from an external quartz on the UFO board. Leakage is measured using a Tektronix CT-1 current probe and sampled at 62.5 MSamples/s by a PicoScope 5244D with 12 bits of resolution.

SKINNY. As mentioned before, we use the optimized 32-bit fix-sliced C-implementation of SKINNY from [AP21] which achieves state-of-the-art performance for binary ciphers on ARM processors. Like in bit-sliced implementations, fix-sliced ones store each bit of a byte in different registers (called slices), but it reduces the number of cycles by avoiding the overhead of the ShiftRows operation which is expensive to compute on a sliced state. For that, the technique instantiates multiple slightly different MixColumns operations and adapts the execution of the SubByte operation, which increases the code size. In addition to fix-slicing, the authors of [AP21] propose a decomposition of the SKINNY 8-bit S-Box from 8 bits to 4 bits, exploiting the symmetry in the design. This decomposition allows filling each register entirely and leads to the use of only 4 registers to pack the whole state instead of 8. In the end, the fix-sliced implementation is about 2.5x faster compared to bit-sliced versions at the expense of doubling the code size. We refer to [AP21] for the full details on the optimizations and performances. To the best of our knowledge, no C-language masked bit/fix-sliced software implementation of SKINNY exists in the public space. Hence, we have built our own (which is contained in our public repository) based on the provided unprotected fix-sliced one. Our masked implementations follow the same architecture, with the linear layer applied independently to each share while the non-linear layer is using PINI gadgets from [CGLS21] to ensure provable security. As mentioned above, in our implementations, the key schedule is masked but the tweak schedule is not as the tweak is considered public.

mid-pSquare. For mid-pSquare we have developed straightforward 32-bit C-language implementations. Since each word of the state and inputs already occupies almost the full data width of registers and buses, there is no need to parallelize operations or change representations (bit-/fix-slicing would anyway not be applicable). The target for the attack is the first squaring operation in the first round which is implemented using the gadget from [CMM+23] that has been proven secure at arbitrary orders.

Noise Level. A quantitative leakage evaluation of the target device is provided in Appendix A based on a simple example computation (not directly related to the field arithmetic used by our target algorithms). We confirm that, as indicated in previous works already [BS21], the physical noise level provided by such Cortex-M microcontrollers is very low and operands that are touched by a few consecutive instructions can be leaked almost in full (at least up the sizes we tested). Hence, without additional countermeasures or assumptions, adversaries with profiling access to the device will not be hindered significantly by the physical noise level, motivating the need for masking schemes which minimize the reliance on noise.

Attack methodology and results. We target the first round of both ciphers without tweak and limit our analysis to masked implementations using $d = 2$, 3 and 4 shares to keep the analysis complexity manageable. The attacks are performed in three steps.

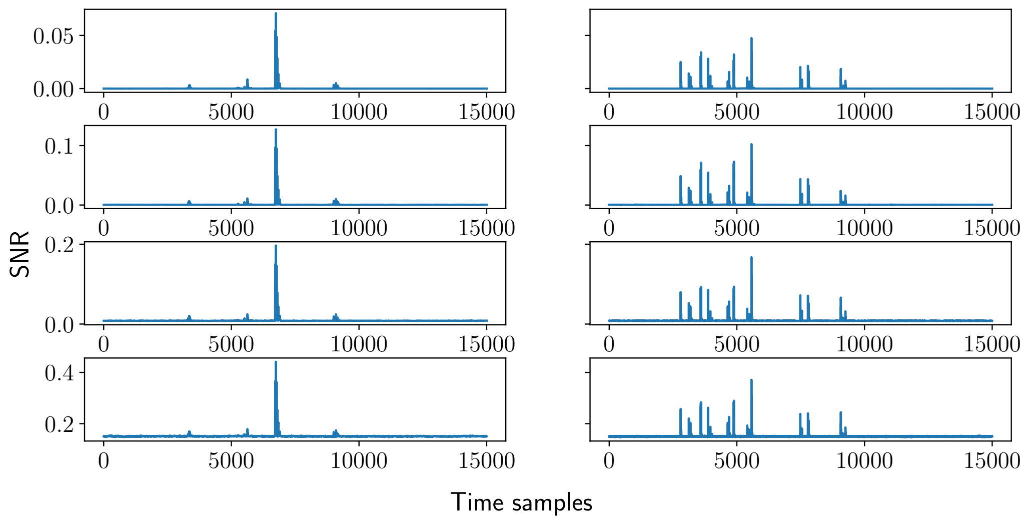


Figure 6: Exemplary SNR figures for mid-pSquare (left) and SKINNY (right) when considering the lower 4, 8, 12 and 16 bits (top to bottom) out of the 31/32-bit values processed by the non-linear layer.

First, a training dataset, where all intermediate values (including fresh randomness) are assumed to be known, is used to create models of the shared inputs and outputs of the non-linear operations (modular squaring for mid-pSquare, S-box for SKINNY). Corresponding SNR values for the lower 4, 8, 12 and 16 bits of a value processed by the non-linear layer are depicted in Figure 6. It can be observed that the maximum SNR which is naturally upper bounded by the ratio of bits included in the computation to the total number of bits in the operand (31 and 32, respectively) is slightly larger for mid-pSquare while more sample points admit a notable SNR peak for SKINNY. We believe that these results are reasonably comparable. The attacker then models the leakage distribution associated to each share.

The tool used to model the leakage distribution is Regression-Based Linear Discriminant Analysis (RLDA) introduced at CHES 2023 [CDSU23] which is particularly effective for handling large ($\geq$ 16 bits) intermediate values. In the second step, the attacker uses an attack dataset, where only the unshared input associated to each trace is known. The independent probability distributions for each share obtained in step one can be combined using a technique called Soft Analytical Side Channel Attack (SASCA) [VGS14]. SASCA enables the attacker to retrieve the probability associated to the unmasked value by propagating the probabilities of each share to its unmasked representation using the encoding relations between the variables. After running the SASCA belief propagation, the attacker gets the probabilities on the unmasked inputs and outputs of the non-linear operation. Then the adversary may infer the probabilities related to each key guess using the known input of the first round. Finally, the attacker combines the probabilities from multiple attack traces that use the same key by assuming independence between the encryption executions. We have performed these three steps using the open-source SCALib library [CB23].

The resulting key ranks for the profiled horizontal SASCA attacks on SKINNY and the reduced versions of mid-pSquare are depicted in Figures 7 and 8, respectively. As expected, the attack on masked fully fix-sliced SKINNY implementations succeeds with a low number of attack traces. In particular a 4th-order attack on the 4-share implementation requires only a few thousand traces to isolate the correct key candidate, which is consistent with [BS21]. For the reduced versions of mid-pSquare, this is only true for the 7-bit implementation. Here the amplification in the prime size appears to be too small to render such attacks hard. Still, it is important to note that the 7-bit implementation achieves this result without the help of algorithmic noise, as only 7 bits of each register/operand are occupied while all remaining ones are zero. SKINNY's SCA security on the other hand benefits from at least half of the state contributing to algorithmic noise. However, as evident from our results increasing the prime size of mid-pSquare while keeping operations and masking order identical increases the hardness of the attack drastically, losing more than two orders of magnitude in attack performance from 7-bit to 19-bit. While we were unable to analyze the attack on the full 31-bit versions[13], we have extrapolated the trends on the reduced version to predict the potential attack success on mid-pSquare. We believe such extrapolation is even conservative considering that the relative amount of information that can be extracted through the leakage typically decreases when increasing the field size (due to operating on more bits in parallel). Assuming that the attack approximation on mid-pSquare is sound, the security gain compared to SKINNY would be almost 5 orders of magnitude large.[14]

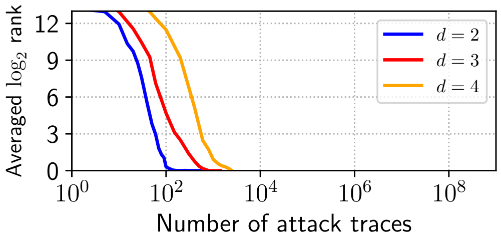


Figure 7: Key rank estimation on SKINNY for different masking orders.

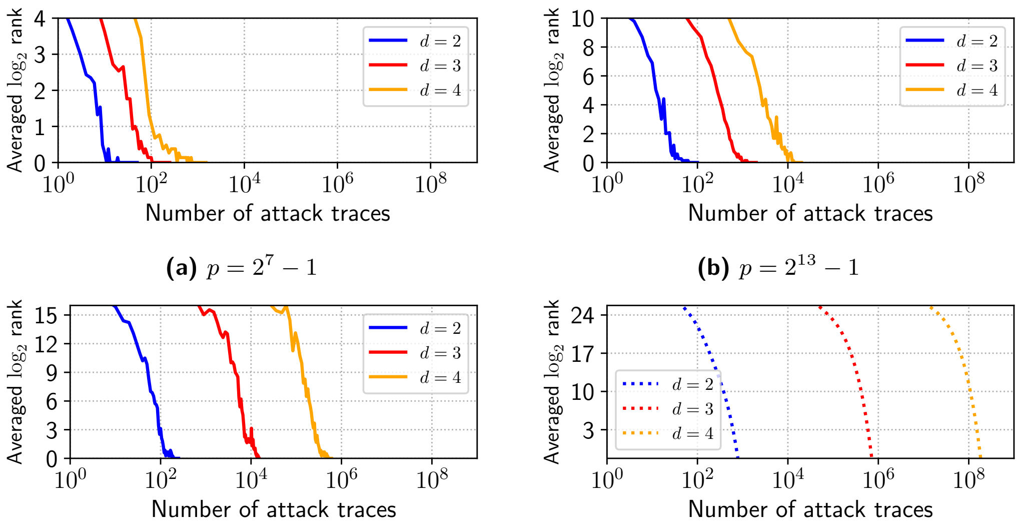


Figure 8: Key rank estimation on the different reduced mid-pSquare instances. The results for mid-pSquare with $p = 2^{31} - 1$ are obtained through extrapolation.

[13] We emphasize the infeasibility of performing even parts of our analysis on the full mid-pSquare. Acquiring the 19-bit results of Figure 8c already took more than a week on a dedicated computation server with 128-thread CPU under full load. We estimate that reproducing the key rank estimation for the full 31-bit version with only 2 shares would take at least 6 months of computation time on the same server. Completing the graph for 3 and 4 shares is therefore estimated to take multiple years.

[14] We recall that we propose mid-pSquare only with a 31-bit modulus. The toy versions in this section are analyzed for illustration and to practically confirm the scaling trends described in [FMM+24].

6 Conclusion

Our novel instance of tweakable lightweight block cipher mid-pSquare, optimized for efficient prime-field masking on 32-bit software platforms, provides excellent performance and security figures compared to the state of the art. It is easy to implement, uses little memory, leads to short code, fast compilation times and small binaries even at very high security orders. Considering that on top of the huge advantages in side-channel security against profiled horizontal attacks (which currently are among the biggest threats to masked software implementations), mid-pSquare is also expected to deliver significantly better fault resistance when masked, right out of the box [MSS24], we believe it is fair to conclude that the design principles of mid-pSquare have great potential to improve the concrete physical security of symmetric cryptography in software in the near future.

Acknowledgments

Lorenzo Grassi was supported by the European Research Council (ERC), grant number 101160608 "SYMPZON". Pierrick Méaux was funded by the ERC under the Advanced Grant program (grant number: 787390). François-Xavier Standaert and Thorben Moos are research director and post-doctoral researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.), respectively. This work has been funded in part by the ERC Advanced Grant number 101096871. Views and opinions expressed are those of the authors 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

[ABB+20] Melissa Azouaoui, Davide Bellizia, Ileana Buhan, Nicolas Debande, Sébastien Duval, Christophe Giraud, Éliane Jaulmes, François Koeune, Elisabeth Oswald, François-Xavier Standaert, and Carolyn Whitnall. A systematic appraisal of side channel evaluation strategies. In *SSR*, volume 12529 of *LNCS*, pages 46–66. Springer, 2020.

[AP21] Alexandre Adomnicai and Thomas Peyrin. Fixslicing aes-like ciphers new bitsliced AES speed records on arm-cortex M and RISC-V. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):402–425, 2021.

[BBC+23] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. New design techniques for efficient arithmetization-oriented hash functions: Anemoi permutations and Jive compression mode. In *CRYPTO (3)*, volume 14083 of *LNCS*, pages 507–539. Springer, 2023.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *EUROCRYPT*, volume 1592 of *LNCS*, pages 12–23. Springer, 1999.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-Order Differential Properties of Keccak and *Luffa*. In *FSE*, volume 6733 of *LNCS*, pages 252–269. Springer, 2011.

[BCD+20] Tim Beyne, Anne Canteaut, Itai Dinur, Maria Eichlseder, Gregor Leander, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Yu Sasaki, Yosuke Todo, and Friedrich Wiemer. Out of Oddity - New Cryptanalytic Techniques Against

Symmetric Primitives Optimized for Integrity Proof Systems. In *CRYPTO*, volume 12172 of *LNCS*, pages 299–328. Springer, 2020.

[BCPZ16] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal Side-Channel Attacks and Countermeasures on the ISW Masking Scheme. In *CHES*, volume 9813 of *LNCS*, pages 23–39. Springer, 2016.

[BFSY05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *Proc. of MEGA*, volume 5, 2005.

[BGG$^+$14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In *CARDIS*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

[BJK$^+$16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *CRYPTO*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

[BL25] Daniel Bernstein and Tanja Lange. eBACS: ECRYPT Benchmarking of Cryptographic Systems. `https://bench.cr.yp.to`, Accessed 10 March 2025.

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.

[BS93] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[BS21] 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):202–234, 2021.

[BV25] Tim Beyne and Michiel Verbauwhede. Integral cryptanalysis in characteristic *p*. Cryptology ePrint Archive, Paper 2025/932, 2025.

[CB23] Gaëtan Cassiers and Olivier Bronchain. Scalib: A side-channel analysis library. *J. Open Source Softw.*, 8(86):5196, 2023.

[CDSU23] Gaëtan Cassiers, Henri Devillez, François-Xavier Standaert, and Balazs Udvarhelyi. Efficient regression-based linear discriminant analysis for side-channel security evaluations towards analytical attacks against 32-bit implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):270–293, 2023.

[CGG$^+$22] Carlos Cid, Lorenzo Grassi, Aldo Gunsing, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. Influence of the Linear Layer on the Algebraic Degree in SP-Networks. *IACR Trans. Symmetric Cryptol.*, 2022(1):110–137, 2022.

[CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware Private Circuits: From Trivial Composition to Full Verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021.

[CGP$^+$12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO*, volume 1666 of *LNCS*, pages 398–412. Springer, 1999.

[CLO13] David Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra.* Springer Science & Business Media, 2013.

[CMM$^+$23] 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):482–518, 2023.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 423–440. Springer, 2014.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021.

[DFS15] 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*, volume 9056 of *LNCS*, pages 401–429. Springer, 2015.

[DFS16] Stefan Dziembowski, Sebastian Faust, and Maciej Skórski. Optimal Amplification of Noisy Leakages. In *TCC (A2)*, volume 9563 of *LNCS*, pages 291–318. Springer, 2016.

[DGGK21] Christoph Dobraunig, Lorenzo Grassi, Anna Guinet, and Daniël Kuijsters. Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields. In *EUROCRYPT*, volume 12697 of *LNCS*, pages 3–34. Springer, 2021.

[DKLS20] Orr Dunkelman, Abhishek Kumar, Eran Lambooij, and Somitra Kumar Sanadhya. Counting Active S-Boxes is not Enough. In *INDOCRYPT*, volume 12578 of *LNCS*, pages 332–344. Springer, 2020.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Information Security and Cryptography. Springer, 2002.

[EGL$^+$20] Maria Eichlseder, Lorenzo Grassi, Reinhard Lüftenegger, Morten Øygarden, Christian Rechberger, Markus Schofnegger, and Qingju Wang. An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC. In *ASIACRYPT, Part I*, volume 12491 of *LNCS*, pages 477–506. Springer, 2020.

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, 2002.

[FGHR14] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. Sub-cubic change of ordering for gröbner basis: a probabilistic approach. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 170–177. ACM, 2014.

[FM17] Jean-Charles Faugère and Chenqi Mou. Sparse FGLM Algorithms. *Journal of Symbolic Computation*, 80(3):538–569, 2017.

[FMM$^+$24] Sebastian Faust, Loïc Masure, Elena Micheli, Maximilian Orlt, and François-Xavier Standaert. Connecting leakage-resilient secret sharing to practice: Scaling trends and physical dependencies of prime field masking. In *EUROCRYPT (4)*, volume 14654 of *Lecture Notes in Computer Science*, pages 316–344. Springer, 2024.

[GHR$^+$23] Lorenzo Grassi, Yonglin Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, and Qingju Wang. Horst meets Fluid-SPN: Griffin for zero-knowledge applications. In *CRYPTO (3)*, volume 14083 of *LNCS*, pages 573–606. Springer, 2023.

[GKR$^+$21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In *30th USENIX Security Symposium USENIX Security 2021*, pages 519–535. USENIX Association, 2021.

[GLR$^+$20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In *EUROCRYPT (2)*, volume 12106 of *LNCS*, pages 674–704. Springer, 2020.

[GMM$^+$24] Lorenzo Grassi, Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert. Generalized Feistel Ciphers for Efficient Prime Field Masking. In *EUROCRYPT*, volume 14653 of *LNCS*, pages 188–220. Springer, 2024.

[GØSW23] Lorenzo Grassi, Morten Øygarden, Markus Schofnegger, and Roman Walch. From Farfalle to Megafono via Ciminion: The PRF Hydra for MPC applications. In *EUROCRYPT*, volume 14007 of *LNCS*, pages 255–286. Springer, 2023.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In *CHES*, volume 6917 of *LNCS*, pages 326–341. Springer, 2011.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: The romulus and remus families of lightweight AEAD algorithms. *IACR Trans. Symmetric Cryptol.*, 2020(1):43–120, 2020.

[JK97] Thomas Jakobsen and Lars R. Knudsen. The Interpolation Attack on Block Ciphers. In *FSE*, volume 1267 of *LNCS*, pages 28–40. Springer, 1997.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Knu94] Lars R. Knudsen. Truncated and Higher Order Differentials. In *FSE*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.

[Lai94] Xuejia Lai. Higher Order Derivatives and Differential Cryptanalysis. In *Communications and Cryptography: Two Sides of One Tapestry*, pages 227–233. Springer US, 1994.

[LR88] Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, 1988.

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[Mas93] James L. Massey. SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm. In *Fast Software Encryption - FSE 1993*, volume 809 of *LNCS*, pages 1–17. Springer, 1993.

[Mat93] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In *EUROCRYPT*, volume 765 of *LNCS*, pages 386–397. Springer, 1993.

[MCHS23] Loïc Masure, Gaëtan Cassiers, Julien M. Hendrickx, and François-Xavier Standaert. Information bounds and convergence rates for side-channel security evaluators. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):522–569, 2023.

[MMMS23] Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert. Effective and Efficient Masking with Low Noise Using Small-Mersenne-Prime Ciphers. In *EUROCRYPT*, volume 14007 of *LNCS*, pages 596–627. Springer, 2023.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards.* Springer, 2007.

[MSS24] Thorben Moos, Sayandeep Saha, and François-Xavier Standaert. Prime masking vs. faults - exponential security amplification against selected classes of attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(4):690–736, 2024.

[Pat98] Jacques Patarin. About Feistel Schemes with Six (or More) Rounds. In *FSE*, volume 1372 of *LNCS*, pages 103–121. Springer, 1998.

[Pat04] Jacques Patarin. Security of Random Feistel Schemes with 5 or More Rounds. In *CRYPTO*, volume 3152 of *LNCS*, pages 106–122. Springer, 2004.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks: A Formal Security Proof. In *EUROCRYPT*, volume 7881 of *LNCS*, pages 142–159. Springer, 2013.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In *CHES*, volume 3659 of *LNCS*, pages 30–46. Springer, 2005.

[SM16] Tobias Schneider and Amir Moradi. Leakage assessment methodology - extended version. *J. Cryptogr. Eng.*, 6(2):85–99, 2016.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[Ste25] Matthias Johann Steiner. Gröbner basis cryptanalysis of Ciminion and Hydra. *IACR Transactions on Symmetric Cryptology*, 2025(1):240–275, Mar. 2025.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft Analytical Side-Channel Attacks. In *ASIACRYPT*, volume 8873 of *LNCS*, pages 282–296. Springer, 2014.

[VXXZ24] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 3792–3835. SIAM, 2024.

[Wag99] David A. Wagner. The Boomerang Attack. In *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

[WO19] Carolyn Whitnall and Elisabeth Oswald. A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 256–284. Springer, 2019.

[YK21] Shih-Chun You and Markus G. Kuhn. Single-trace fragment template attack on a 32-bit implementation of keccak. In *CARDIS*, volume 13173 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2021.

A Quantitative Leakage Evaluation

To understand the leakage behavior of our target device (an STM32F415RGT6 ARM Cortex-M4 microcontroller, more details in Section 5), we exemplarily analyze the maximum amount of information that can be extracted from a sample computation when there is no algorithmic noise. The latter requirement means that registers and operands are filled by values large enough to be profiled in full, while the remaining bits are zero. We have chosen a simple $x \mapsto x^5$ power map computation over $\mathbb{Z}_{2^n}$ for flexible operand width n, which consists of three multiplications and is compiled without optimizations (`-O0`). The corresponding power trace and SNR are depicted in Figure 9.[15] The three peaks for the multiplications are clearly visible.

After selecting the points of interest based on the SNR peaks, we perform Linear Discriminant Analysis (LDA) using the SCALib library [CB23] to model the leakage with 20 dimensions for $n = 4$, 6, 8, 10 and 12. We then compute the metrics Perceived Information (PI) and Training Information (TI) in function of the number of training traces and plot their convergence in Figure 10.

The PI is a lower bound on the Mutual Information (MI), which constitutes the metric we are actually interested in but which cannot be computed if the real leakage distribution is unknown [MCHS23] (as is always the case in experimental studies). The TI then gives an upper bound on the learnable information for parametric leakage models [MCHS23]. We observe that the simple power map computation consisting of only three multiplications leaks almost in full about the n input bits, e.g., the PI is > 9 and ≈ 11 bits for $n = 10$ and $n = 12$, respectively. This signifies highly leaky computations and a very low physical noise level, motivating the use of prime-field masking which is known to provide superior security guarantees in this context [DFS16, MMMS23].

[15] We also tested such experiments over $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^n-1}$ and obtained similar results.

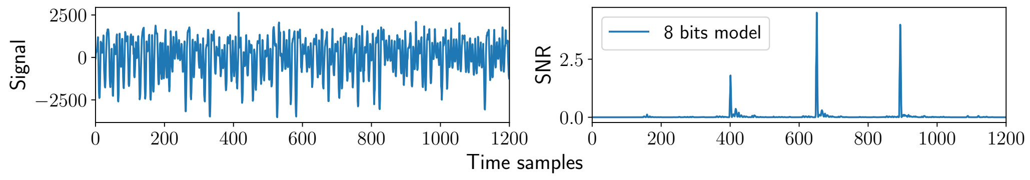


Figure 9: Power trace (left) and 8-bit SNR (right) of exemplary power map computation to assess the available amount of leakage on the target device.

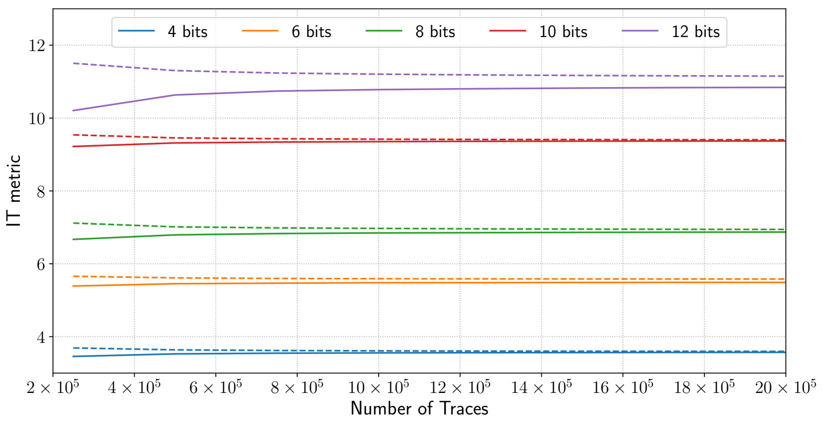


Figure 10: PI (solid lines) and TI (dashed lines) convergence of the LDA-based templates.

B Transitional Leakages

While profiled horizontal attacks are currently assumed to be most effective attack vector on masked software implementations, especially bit-sliced ones [BS21], it is fair to wonder whether (potentially univariate) transitional leakages from microarchitectural overwriting could not change the picture. To investigate this question, we first perform an information theoretic analysis of the maximum amount of information an adversary can possibly extract from the noise-free Hamming distance (assumed to be leaked through overwriting) between two shares in a first-order masked implementation.

The results, depicted in Figure 11, show that the leakage is much smaller in the case of additive prime-field masking compared to Boolean masking. While simple additive masking over prime fields is clearly not immune to transitional leakages, it can be observed that not only the relative amount of information leaked decreases in the field size in case of prime-field masking, but even the absolute one. This highlights that transitional leakages are generally expected to be less of an issue for prime-field masked implementations and once again confirms that increasing the field size is an effective tool for improving the concrete side-channel security guarantees provided by prime-field masking.

To connect this information theoretic analysis to practical observations, we also performed a leakage detection experiment. Figure 12 depicts Test Vector Leakage Assessment (TVLA) evaluations of first-order masked `SKINNY` and mid-pSquare implementations measured on the same target platform as introduced before. While transitional leakages appear to lead to security order reductions in both cases, the difference in magnitude and number of traces required for detection confirm experimentally that these leakages are of lesser extent for masked mid-pSquare. We must stress, however, that simple changes to the source code (e.g., switching the outer and inner *for* loops used for tweakey additions) or compiler optimization flags (e.g., switching `-O3` to `-O1`) can have a significant impact on

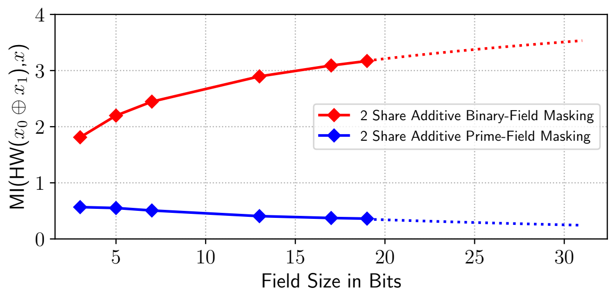
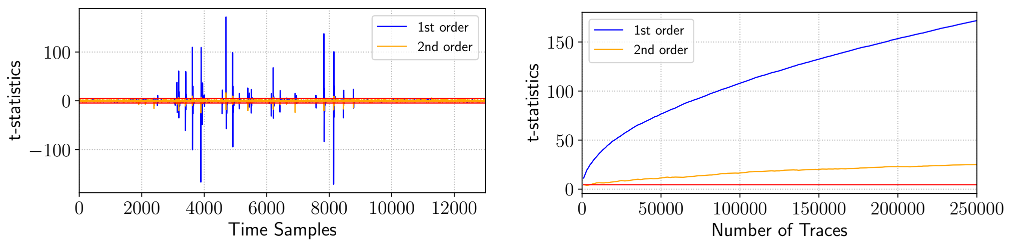
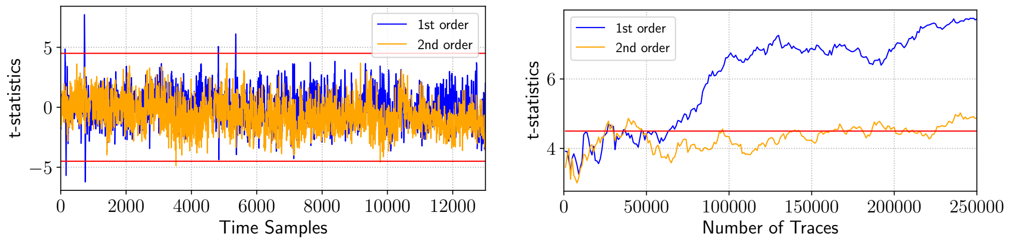


Figure 11: Information theoretic evaluation of the transitional leakage caused by overwriting, assuming (simulated) Hamming distance leakage between two shares.

the magnitude of the maximum t-statistics value for a given number of measurements in Figure 12. In general, our work does *not* aim at guaranteeing the independent leakage assumption or passing common leakage detection tests with the smallest possible number of shares, which inevitably requires a closer inspection of the concrete microarchitecture of the target device. Such issues have already been tackled in a large number of contemporary works on secure masking in software and, if needed, the same principles can be applied to mid-pSquare. Yet, in all tested scenarios, the leakage detection tests required a significantly smaller amount of traces to overcome the detection threshold for the experiments on masked SKINNY compared to masked mid-pSquare on the same target device.



(a) SKINNY-128-128 2-Share implementation



(b) mid-pSquare ($\tau = 0$) 2-Share implementation

Figure 12: Left: TVLA over points, right: TVLA over traces (for mid-pSquare and SKINNY).

C Additional Details on Gröbner Basis Attacks

C.1 Linearization With Related-Tweaks

To the best of our knowledge, related-tweak Gröbner basis attacks are unknown in the literature. In the following, we argue that a related-tweak Gröbner basis attack on mid-pSquare reduces to a slightly improved two-plaintext-ciphertext attack. Let $\mathbf{p}_1, \mathbf{p}_2 \in \mathbb{F}_p^4$ be two plaintext samples encrypted under the same key with tweaks $\mathbf{t}_0, \mathbf{t}_1 \in \mathbb{F}_p^4$.

We denote $\boldsymbol{\Delta t} = \mathbf{t}_0 - \mathbf{t}_1$, and let $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ denote the intermediate state variables in the models of $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively. In a two-plaintext-ciphertext attack, we can generate a new linear polynomial by computing the difference between quadratic polynomials from the first round. Explicitly, this yields:

$$\begin{aligned} &\underbrace{(k_0 + p_{1,0} + t_0)^2 + x_0^{(1)} + p_{1,2}}_{= f_{\mathbf{p}_1,0}^{(1)}} - \underbrace{(k_0 + p_{2,0} + t_0 + \Delta t_0)^2 - y_0^{(1)} - p_{2,2}}_{= f_{\mathbf{p}_2,0}^{(1)}} \\ &= x_0^{(1)} - y_0^{(1)} + 2 \cdot k_0 \cdot (p_{1,0} - p_{2,0} - \Delta t_0) + \left(p_{1,0} + t_0^2\right)^2 - (p_{2,0} + t_0 + \Delta t_0)^2, \end{aligned} \tag{11}$$

where we deliberately absorbed the Feistel constants into $\mathbf{t}_0$ and $\mathbf{t}_1$. If $p_{1,0} - p_{2,0} + \Delta t_0 \neq 0$, then this linear polynomial has three terms, and we are in the same scenario as in a two-plaintext-ciphertext attack. On the other hand, with related tweaks the adversary could enforce that $p_{1,0} - p_{2,0} + \Delta t_0 = 0$ which implies that $y_0^{(1)} = x_0^{(1)} - \left(p_{1,0} + t_0^2\right)^2 + (p_{2,0} + t_0 + \Delta t_0)^2$. In addition, by computing $f_{\mathbf{p}_j,1}^{(1)} \mod \left(f_{\mathbf{p}_j,0}^{(1)}\right)$ for both $\mathbf{p}_1$ and $\mathbf{p}_2$, we produce two quadratic polynomials analogous to the construction of the *degrevlex* basis for a single sample. Let us now examine the difference between the quadratic components of $f_{\mathbf{p}_1,1}^{(1)} - f_{\mathbf{p}_2,1}^{(1)} \mod \left(f_{\mathbf{p}_1,0}^{(1)}, f_{\mathbf{p}_2,0}^{(1)}\right)$, which is given by:

$$\left(x_0^{(1)} - k_0 - k_2\right)^2 - \left(y_0^{(1)} - k_0 - k_2\right)^2 = x_0^{(1)^2} - y_0^{(1)^2} + 2 \cdot (k_0 + k_2)\left(y_0^{(1)} - x_0^{(1)}\right). \tag{12}$$

(The quadratic components are derived as in Equation (6).) Thus, if the adversary enforces that $p_{1,0} - p_{2,0} + \Delta t_0 = 0$, then he can linearize another quadratic polynomial for free. Therefore, in a related-tweak attack the adversary can improve the semi-regular Hilbert series for two samples to:

$$\mathsf{H}(z) = \frac{\left(1 - z^2\right)^{4 \cdot r - 3}}{(1 - z)^{4 \cdot r - 7}}. \tag{13}$$

For example, for $r = 16$ (and $k = 2$), the degree of regularity is $D_{\text{reg}} = 24$, and with $\omega = 2$ the (adjusted) binomial coefficient evaluates to approximately 135 bits.

As a result, we conjecture that a related-tweak Gröbner basis attack only mildly improves beyond the capabilities of a two-plaintext-ciphertext attack.