Power Analysis and Countermeasures on the MiMC Block Cipher

Elena Andreeva[1], Stefan Mangard[2], Rishub Nagpal[2], Arnab Roy[3] and Stefano Trevisani[1]

[1] TU Wien, Vienna, Austria, name.surname@tuwien.ac.at
[2] TU Graz, Graz, Austria name.surname@tugraz.at
[3] Universität Innsbruck, Innsbruck, Austria arnab.roy@uibk.ac.at

Abstract. Modern zero-knowledge (ZK), fully homomorphic encryption (FHE) and Multi-party Computation (MPC) protocols have motivated research interest in Arithmetization-Oriented (AO) cryptographic primitives. The use of these protocols on embedded platforms requires consideration for protection against side-channel analysis (SCA), including timing and power attacks. Compared to traditional bit-oriented block ciphers, the design of side-channel countermeasures for AO-based ciphers poses unique challenges due to their different mathematical properties and computation requirements.

In this work, we perform side-channel analysis of MiMC, a well-known AO block cipher. We first consider its constant-time implementation over the BN254 prime field for both x86 and ARM-v7 targets. Next, we demonstrate both profiled (SASCA) and unprofiled (linear regression) power analysis of our implementation on the ARM Cortex-M4 microcontroller, showing that a DPA adversary can reduce the key-guessing space to just 2^{30} candidates using only $\approx 32\,000$ power measurements, and the profiling adversary can perform full key recovery in as few as 100 measurements.

Hence, we consider and compare two side-channel countermeasures: a classical ISW masking approach, and the redundant number representation (RNR) adapted for large prime fields. For the RNR countermeasure, we show that ≈ 64 bits of redundancy are sufficient to reduce the information leakage enough to make DPA attacks impractical. Our claim is supported both by a standard fixed-vs-random leakage assessment with 10 million collected traces and by quantitative analysis via SASCA attacks. Finally, we compare the performance of the RNR countermeasure with first-order masking. While masking incurs in a $\approx 8.3\times$ overhead on a Cortex-M4 MCU and in an $\approx 81\times$ overhead on an x86-64 CPU, the RNR approach only introduces a $\approx 50\%$ overhead on both targets.

Keywords: Power Analysis, Arithmetization-Oriented, MiMC, block ciphers

1 Introduction

Privacy-enhancing protocols such as zero-knowledge proofs (ZKP) [Gro16, BSBHR18, GWC19], multi-party computing (MPC) [DKL+13, KOS16] and (Fully-)homomorphic encryption (FHE) [Gen09, BV11, CKKS17], enable secure computation without disclosing private data beyond access restriction. Such protocols are then particularly attractive for Internet-of-Things (IoT) applications, as IoT devices and services are already and increasingly more pervasive, and the amount of data collected, transmitted and elaborated, with the support of increasingly faster communication and cloud infrastructures, keeps growing [QWA+19]. In order to ensure the confidentiality, integrity and authenticity of sensitive data, both during collection, transmission and processing, a variety of cryptographic solutions are required [DSSR24, Moz]. For example, data transmission can be

secured by means of authenticated encryption (AE) schemes, and computations involving sensitive data can be made private by running the computation within an MPC protocol. While there have been advancements in executing privacy-enhancing protocols directly on embedded devices [SD21, PGS23], a more scalable solution is for the devices to send encrypted data to a cloud backend, and then run an MPC protocol, over the encrypted data, in the cloud [BPA+23]. While the exact cost is highly protocol-dependant, a good approximation metric is often given by the multiplicative complexity of the computation, when intended as an arithmetic circuit over the protocol's native algebraic field $\mathbb{F}_q$ (where either $q = 2^k$, or q is prime). Hence, employing algorithms with low multiplicative complexity over $\mathbb{F}_q$ is paramount to ensure the efficiency of the protocol's execution. This is true in particular for computations over encrypted data, where the encryption and decryption process should be as efficient as possible even when evaluated in-protocol.

Arithmetization-Oriented (AO) cryptography studies the design of cryptographic primitives (mainly block ciphers and hash functions) that have a low multiplicative complexity when instantiated within a given ZKP/MPC protocol [AABS+19, GKR+21]. MiMC [AGR+16], is the earliest such AO block cipher, and its core design, lies at the hearth of many newer AO primitives [GLR+19, GKR+21, GKS23]. Fundamentally, MiMC relies on an iterated construction whose round function consists of the application of a low-degree monomial, over a suitable finite field (usually a prime field $\mathbb{F}_p$), to the sum of the plaintext with the round key (see Figure 1 for a visual depiction). Its elegant algebraic description, and its low multiplicative complexity over the field of definition, have made MiMC a popular choice for both MPC and ZKP applications, either instantiated as a block cipher or a hash function [AGP+19, SD21, ASM+24]. Another important aspect of MiMC is its resistance to a wide class of cryptanalytical techniques. Indeed, its design has been shown to effectively withstand several cryptanalytical attacks, both statistical and algebraic [LP19, EGL+20, RAS21].

However, when dealing with IoT embedded devices, attacks based on side-channel analysis (SCA) become a serious threat. Specifically, power analysis techniques, such as Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [KJJ99, MOP07], can be particularly concerning. If we are to use MiMC to encrypt data on an embedded device, it is then crucial to ensure that its implementation is able to withstand SCA attacks.

One of the standard countermeasures proposed in the literature for protecting a block cipher against power-based SCA is to employ a masking scheme based on secret-sharing, such as the ISW scheme [ISW03]. Masking is a well-understood countermeasure, and (arithmetic) masking seems like a perfect match for MiMC, due to it being specifically designed to have a simple arithmetic circuit representation. Still, masking can introduce a significant overhead, due to the large amounts of randomness required to refresh masks when performing multiplications.

An interesting alternative to masking for primitives working over finite rings/fields of integers, is the Redundant Number Representation (RNR). The RNR countermeasure has been successfully applied in the past to protect lattice-based cryptographic primitives with a relatively small overhead [ZBT19, HP23b, NHPM25]. The approach of RNR is radically different from masking: in order to hide a secret s over $\mathbb{F}_p \cong \mathbb{Z}_p$, we encode it as a random representative of its equivalence class, that is, as an integer $s + rp$ for some random $0 \leq r < \eta$. Arithmetic is then performed over $\mathbb{Z}_{\eta p}$, and the final result can be reduced to canonical representation when needed. The lattice-based primitives over which the RNR countermeasure was applied, are all operating on small rings, in which addition and multiplication can be fully done within a single 32-bit register. However, a typical MiMC instance is defined over a ≈ 256-bit prime field. Unlike for masking, the security provided by the RNR countermeasure is not understood as well, and hence a careful leakage analysis is essential to assess its effectiveness for a given choice of η.

1.1 Contributions

In this work, we tackle the problem of implementing the MiMC-p block cipher securely against timing and power side-channel attacks. Concretely:

- We devise an efficient, constant-time implementation of MiMC over the BN254 elliptic curve scalar field, which we chose both as it guarantees cryptanalytical security, and due to its popularity especially in the ZKP space [ALNS13, BR17]. However, we stress right away that our analysis may be easily conducted over other suitable choices of primes. Our implementation runs on both x86-64 and ARM platforms, although we are mostly interested in the latter, especially as Cortex microcontrollers are a more standard target for SCA. When comparing different implementation strategies, we conclude that the CIOS Montgomery multiplication is the best approach, being up to $\approx 20\%$ faster than Barrett multiplication, and allowing us to 'skip' conditional subtractions. On ARM microcontrollers, we validated constant-timeness both by analyzing the generated assembly and collecting execution traces with a ChipWhisperer capture board. On x86 targets, we instead relied on a statistical assessment carried out with the `dudect` tool [RBV17].

- We describe two power analysis attacks on MiMC, one based on unprofiled linear regression and the other based on soft-analytical side-channel analysis (SASCA). Our attacks, which target the key addition step of the first round of MiMC encryption, exploit the leakage that is observed due to the carry propagation induced by modular addition. In the unprofiled case, our attack is able to reduce the key guessing entropy from 254 bits to 31 bits after 32 000 traces. In the profiled case, our attack recovers the full key in only 100 traces.

- We propose the shuffled redundant number representation (RNR) countermeasure [HP23a, Reg05] as our primary recommendation to protect MiMC against power analysis. While this approach is well-known in the context of RSA and ECC [SOP08], to the best of our knowledge this is the first time this countermeasure is applied to protect a block cipher over a large finite field. We argue the effectiveness of the RNR countermeasure, providing a formal bound for Mersenne primes and an empirical bound for unstructured primes.

- Finally, we give practical parameters to obtain a secure shuffled RNR implementation of MiMC. We perform a test-vector leakage assessment (TVLA) on the STM32F3 target, showing that the RNR countermeasure does not exhibit data dependant leakage at least up to 10 million power measurements. Furthermore, we perform SASCA attacks for different RNR implementations to analyze the countermeasure's effectiveness against profiling adversaries. We then conclude by comparing the performance of an RNR implementation of MiMC with first-order masking. While masking incurs in a $\approx 8.3\times$ overhead on a Cortex-M4 MCU and in an $\approx 81\times$ overhead on an x86-64 CPU, the RNR approach only introduces a $\approx 50\%$ overhead on both targets.

2 The MiMC Block Cipher

Throughout this work, we denote numeric variables in lowercase, sets in uppercase, tuples in bold, and functions either in lowercase or uppercase. For a set S and a natural number n, let S^n denote the set of n-tuples over S. For any natural number n, let $\mathbb{Z}/n\mathbb{Z}$ be the ring of residue classes modulo n, and $\mathbb{Z}_n$ the ring of integers modulo n, whose elements are the canonical representatives of the equivalence classes in $\mathbb{Z}/n\mathbb{Z}$. Given a prime number p, let $\mathbb{F}_p \cong \mathbb{Z}/p\mathbb{Z}$ be the unique (up to isomorphism) finite field of order p (a prime field).

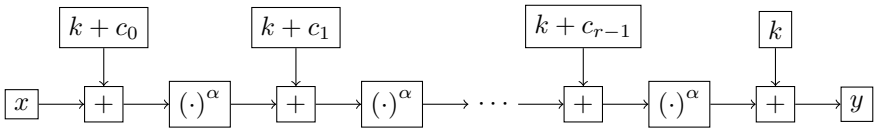


Figure 1: Depiction of the MiMC-p block cipher, all operations are intended modulo p.

Hence, on a digital binary machine, $\mathbb{F}_p$ elements can be encoded as the elements of $\mathbb{Z}_p$ in their base-2 representation, possibly zero-extended to match the machine's word length w.

A block cipher is a family of permutations (bijections) over some message space each named by a key over some key space. We say that a block cipher E is arithmetization-oriented (AO), with respect to some finite field $\mathbb{F}$, if $E = (E_{\boldsymbol{k}} \colon \mathbb{F}^n \to \mathbb{F}^n \mid \boldsymbol{k} \in \mathbb{F}^\kappa)$ for some positive integers κ and n. In the remainder, we will focus our attention on prime fields, that is $\mathbb{F} = \mathbb{F}_p$.

MiMC is an arithmetization-oriented block cipher that was proposed by Albrecht et al. [AGR+16] in 2016. Inspired by the $\mathcal{PURE}$ cipher [NK95, JK97], it was designed to withstand linear and differential cryptanalysis, as well as algebraic attacks (e.g. interpolation attacks and GCD attacks). Its MiMC-p variant uses the prime field $\mathbb{F}_p$ as both its key space and message space. While in its original specification, it was required that $\gcd(3, p-1) = 0$ to make MiMC-p well-defined, this restriction was then lifted in follow-up works [AGR+16, AGP+19, GLR+19]. Hence, we can define the MiMC-p block cipher as follows:

Definition 1 (MiMC-p block cipher Figure 1)**.** Given a prime number p, let the MiMC-p round function be $F_{p,k,c}(x) = (x + k + c)^\alpha$, where $x, k, c \in \mathbb{F}_p$ and α is the smallest integer such that $\alpha \geq 2$ and $\gcd(\alpha, p-1) = 1$. Given a number of rounds r, and constants $c_0, \ldots, c_{r-1} \in \mathbb{F}_p$, the r-round MiMC-p block cipher is then:

$$\text{MiMC-}p_{k,c_1,\ldots,c_r}(x) = \big(F_{p,k,c_{r-1}} \circ \cdots \circ F_{p,k,c_0}\big)(x) + k. \tag{1}$$

Note that, in order to unambiguously define the block cipher, we must specify how the round constants are derived. While the MiMC proposal suggests generating the constants pseudo-randomly [AGR+16], it is not specified exactly how, leaving some freedom to the implementation. We write MiMC-$p_{k,r}$ to leave the choice of round constants implicit in the definition. Further, we let MiMC-p_k to implicitly assume $r = \log_\alpha(p)$, i.e. the recommended number of rounds to ensure a security level of roughly $\lfloor \log_2(p) \rfloor$ bits. Finally, we write MiMC_k when the choice of p is clear from the context, and we equivalently write $\text{MiMC}(k, x)$ to intend $\text{MiMC}_k(x)$.

A graphical depiction of the MiMC-p block cipher is shown in Figure 1. Coprimality of α with $p-1$ guarantees that the associated power map, and consequently the whole function, is bijective for any choice of the key. Several works have been published on the cryptanalysis of MiMC (and variations thereof), and no major weaknesses were found in the block cipher [AGP+19, Bon19, EGL+20, RAS21, BBLP22]. In particular, choosing p such that $\log_2(p) \approx \lambda$ bits provides about λ bits of security against all known attacks. So, for example, choosing a ≈ 256-bit prime gives about ≈ 256 bits of security.

2.1 Target instance

While many choices of p are possible for a certain security level, some have become more relevant than others in practice due to the prime underlying the construction popular elliptic curves, that are used particularly in ZK-SNARKs [ALNS13, BR17, HBHW22]. One of such primes is the BN254 curve scalar field order [Wan22], a 254-bit prime. We hence focus our attention on the implementation of MiMC-p over this particular prime;

Algorithm 1 The MiMC-p block cipher. Note that, over the BN254 scalar field, we have $\alpha = 5$ and $r = 110$.

```
function MIMC(k ∈ F_p, x ∈ F_p)
    y_0 ← x + k
    for i ∈ (1, ..., r − 1) do
        y_i ← (y_{i−1} + k + c_i)^α
    y_r ← y_{r−1} + k
    return y_r
```

our work can be used as a template to study the behavior of implementations over other prime choices.

Following Definition 1, we find that the smallest positive integer coprime with $p - 1$, excluding the unit, is $\alpha = 5$. Hence, the non-linear power map have degree 5, and the number of rounds is going to be $r = \lceil \log_5(p) \rceil = 110$. As we mentioned earlier, there is no strict specification on how round constants should be chosen, except that the process should be 'pseudo-random'. Following a nothing-up-my-sleeves approach, we decided to select the round constants by rejection-sampling over 256-bits blocks, after truncation to 254 bits, generated from the SHAKE256 XOF [Dwo15] with a fixed nonce. A pseudocode description is given in Algorithm 1.

3 Constant Time implementation of MiMC

It is well-known that secret data-dependent code paths or memory accesses can be exploited for timing-based side-channel attacks [BM06]. Thus, it is imperative to utilize constant-time programming techniques when designing side-channel secure software.

To implement MiMC in constant-time, we first choose a suitable representation of the field elements. In our preliminary study, we considered two representations: (i) the standard integer representation, which allows for fast reduction via Barrett division [Bar87] and (ii) the Montgomery representation [Mon85], which allows for fast reduction via Montgomery multiplication. Both representations, and the associated algorithms, are suitable for efficiently performing finite field arithmetic on most instruction set architectures. The computing device can natively manipulate w-bit *words* as elements of the ring $\mathbb{Z}_{2^w}$, and in particular, it can compute addition with carry and double-width multiplication between words. We will assume all these operations to be constant-time, but different instructions may still have different execution times (as is the case on modern CPU microarchitectures). In both standard and Montgomery representations, field elements are represented as n-word tuples, where $n = \lceil \log_2(p)/w \rceil$, where the i-th word represents a base-2^w digit with weight 2^{wi}.

The implementation of field additions is similar for both standard and Montgomery representations. For the key and round-constant additions in Algorithm 1, we use standard "schoolbook" long addition and apply a conditional subtraction of the order p in case of overflow. Since this overflow "fix" is data-dependent, we implement the conditional subtraction as described in Algorithm 2.

For the α-power map, we consider the optimal addition chain of the exponent [Knu98, §4.6.3]: for $\alpha = 5$, evaluating $x^5 = x\big(x^2\big)^2$ requires one multiplication and two squarings.

3.1 Multiplication in the Standard Representation

In the standard representation, every element of $\mathbb{F}_p$ is uniquely represented by an integer in $\{0, \ldots, p - 1\}$. We can hence implement field multiplication as standard integer multiplication followed by reduction modulo p. Since taking the remainder is an expensive and

Algorithm 2 Constant-time small overflow fix.

function FIX_SMALL_OVERFLOW($x \in \{0, \ldots, 2p-1\}$)
  $b \leftarrow 0$
  for $i \in (0, n-1)$ **do**
    $(b, t_i) \leftarrow p_i - x_i - b$
  $m \leftarrow -b \bmod 2^w$ ▷ *all-ones if $b = 1$, else all-zeros*
  $c \leftarrow 0$
  for $i \in (0, n-1)$ **do**
    $(c, y_i) \leftarrow t_i + (p_i \,\&\, m) + c$ ▷ *& represents a bit-wise AND operation.*
  return y

a variable-time operation, we employ the well-known Barrett [Bar87] reduction algorithm. Barrett's reduction algorithm is based on the equivalence:

$$q = \left\lfloor \frac{z}{p} \right\rfloor = \left\lfloor \frac{z2^b}{p2^b} \right\rfloor \approx \tilde{q} = \left\lfloor \frac{z\mu}{2^b} \right\rfloor, \tag{2}$$

where $\mu = \lfloor 2^b/p \rfloor$ can be precomputed given p and $b \geq \max_z\{\lceil \log_2(z) \rceil\}$, and division by 2^b is a simple bit-shift. In particular, it can be shown that $q \leq \tilde{q} \leq q + 1$, and hence the approximate remainder $\tilde{r} = z - \tilde{q}p$ is such that $\tilde{r} \in \{r, r + p\}$, where $r = z - qp$ is the real remainder.

By first precomputing the constant μ for $b = 2\lceil \log_2(p) \rceil$, applying the Barrett reduction algorithm to the standard integer product $z = x \cdot y$, followed by a conditional subtraction of p, we obtain the final constant-time field multiplication algorithm.

3.2 Multiplication in the Montgomery Representation

Algorithm 3 CIOS Montgomery multiplication. An element $x \in \mathbb{F}_p$ is represented as a sequence of w-bit words $x_0, \ldots, x_{n-1}$. Word-wise addition returns a carry bit, and word-wise multiplication computes the full product.

function CIOS_MULTIPLY($x \in \mathbb{F}_p$, $y \in \mathbb{F}_p$)
  $\tilde{p} = -1/p \bmod 2^w$ ▷ *Pre-computed as a constant*
  $z_0, \ldots, z_{n+1} \leftarrow 0, \ldots, 0$
  for $i \in (0, \ldots, n-1)$ **do**
    $c \leftarrow 0$ ▷ *c will store the carry*
    for $j \in (0, \ldots, n-1)$ **do**
      $(c, z_j) \leftarrow z_j + x_j y_i + c$ ▷ *result split in high/low words*
    $(z_{n+1}, z_n) \leftarrow z_n + c$
    $c \leftarrow 0$
    $\mu \leftarrow z_0 \tilde{p} \bmod 2^w$
    $(c, t) \leftarrow z_0 + \mu p_0 + c$
    for $j \in (1, \ldots, n-1)$ **do**
      $(c, z_{j-1}) \leftarrow z_j + \mu p_j + c$
    $(c, z_{n-1}) \leftarrow z_n + c$
    $z_n \leftarrow z_{n+1} + c$
  return FIX_SMALL_OVERFLOW(z) ▷ *At this point $0 \leq z < 2p$*

In order to obtain efficient modular multiplication routines, Montgomery representation [Mon85] relies on the observation that, given some integer z, computing $z/\mu \bmod p$, where $\mu = \left(2^b - p\right) \bmod p$ for some $b \geq \lceil \log_2(p) \rceil$, is easy if we are given the value $\tilde{p} = 1/p$

Table 1: Performance, in clock cycles, of the constant time implementations of various multiplication routines over the BN254 scalar field.

Platform	Barrett mul.	Montgomery mul.	Montgomery sqr.
Core i9-13900KF @ 3.0 GHz	96	79	76
STM32F4 @ 7.37 MHz	1861	1666	1499

mod 2^b. At a high level, it holds that

$$\left\lfloor \frac{z - (z\tilde{p} \bmod 2^b)p}{2^b} \right\rfloor \equiv r/\mu \pmod{p}. \tag{3}$$

Similarly to Barrett reduction, when $b = \lceil \log_2(p) \rceil$, a canonical representation can be obtained after applying a conditional subtraction of the modulus p. Hence, by changing the representation of the field elements via the isomorphism $x \mapsto \mu x$, we can implement field multiplication in terms of a standard long multiplication followed by Montgomery reduction, since $\mu x \cdot \mu y = \mu^2(xy)$ gets reduced into $\mu(xy)$.

There are several known algorithms to implement Montgomery multiplication [BM17, KKAK96], mainly depending on how the long multiplication and reduction steps are interleaved. In particular, the *Coarsely Integrated Operand Scanning* (CIOS) approach is generally considered to be the most efficient, due to its higher data locality and low memory overhead. We give a high-level description of the CIOS Montgomery Multiplication algorithm in Algorithm 3.

Since all the loop bounds are data-independent and known a priori, we can fully unroll Algorithm 3. Hence, again under the assumption that the underlying word-wise operations are constant-time, the resulting field multiplication algorithm will also be constant time.

3.2.1 Delaying the Overflow Fix

Since field elements are stored as n-word tuples, for a total of $nw > b$ bits, Bos and Montgomery showed [BM17] that Montgomery multiplication will give a correct result, up to a conditional subtraction, even when the inputs are in the 'small overflow' range, as long as $4p < 2^{nw}$. For our field choice, we can apply this optimization without needing additional words, since $4p$ fits within the 256 bits corresponding to the total bits in the multi-word representation.

When considering MiMC, this means that, in each round, we can delay resolving the overflow caused by the addition of the key to the plaintext until the end of the round. Although the purpose of this optimization was originally intended purely for performance reasons, it does also have an impact on the power leakage, as even in its constant-time implementation, the small overflow fix can exhibit different power consumption depending on whether we actually subtract p or 0.

3.3 Implementation Details

We implemented the target MiMC-p instance, and the underlying algorithms previously discussed, in the `C` language, making it portable on both ARM and x86 instruction set architectures. However, in order to ensure that no variable-time instructions were generated for the target platforms, we required using platform-specific intrinsics and inline assembly for some of the fundamental routines. The final implementation of MiMC was then obtained by combining the Montgomery arithmetic routines previously discussed as per Algorithm 1. For our x86 test platform, an Intel Core i9-13900KF CPU, we compiled and assembled the source code using the GNU C Compiler Toolchain 15.2.1, at `-O3` optimization level and targeting the Raptor Cove microarchitecture (`-march=raptorlake`). For our ARM test

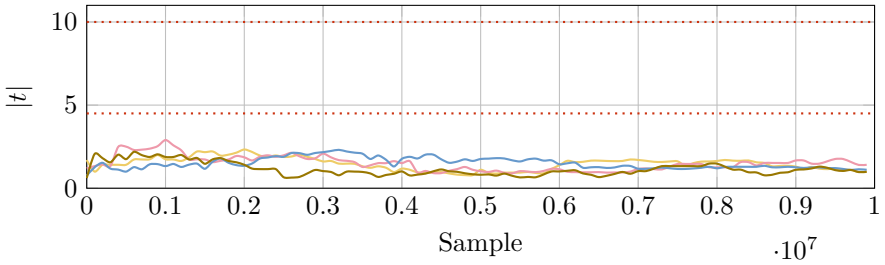


Figure 2: Evolution of the t-statistic reported by `dudect` for the constant-time implementation of MiMC, for various combinations of fixed key/plaintext inputs. All the plots lie well below both the standard critical threshold $\bar{t} = 10$ used by the tool, and the more standard conservative threshold $\bar{t} = 4.5$.

platform, an STM32F4 MCU, we used the GNU ARM Toolchain 14.2.0 for compilation, again at the `-O3` optimizations level, and targeting the ARM Cortex-M4 microarchitecture (`-march=armv7e-m`).

In Table 1, we compare the performance of the Montgomery and Barrett multiplication algorithms for the BN254 scalar field that we implemented for the target architectures.

For the x86 target, the reported cycle count corresponds to the minimum, divided by the batch size, over 100 batches of 2^{20} operations over uniformly distributed inputs. Cycle counts were obtained by querying the time stamp counter on a pinned P-core clocked at 3.0 GHz with simultaneous multi-threading, boosting and address randomization disabled, with the system under minimal load and the process priority set to the highest. For the ARM target, the reported cycle count corresponds to the length of a power trace of the execution collected by a ChipWhisperer CW313 capture board providing an external clock at 7.37 MHz and sampling synchronously at a matching sample rate. We can see that, in both cases, CIOS Montgomery implementation is the best performing algorithm, being $\approx$ 10–20% faster than the Barrett variant. Hence, even when inputs are provided in standard representation, converting in and out of Montgomery form is still going to be advantageous given the large amount of multiplications that are needed to evaluate a full instance of MiMC.

3.3.1 Assessing Constant Time Execution

There are many existing tools that have been proposed in the literature for assessing whether some piece of code executes in constant-time: some tools operate on a high-level description of the code, while others employ statistical analysis over collected measurements. For a recent survey on some of the existing tools, see [FBJ+24]. On the x86 target, we employed the `dudect` leakage assessment tool [RBV17] for our analysis: its statistically-driven approach allows us to easily assess the timing-related leakage of a concrete implementation. We set up `dudect` as follows: after fixing an encryption key, the tool attempts to determine, via a Welch's t-test [WEL47], whether there is a significant difference between the observed timing required to encrypt a series of random field elements as opposed to encrypting some fixed element. In particular, we considered different key-plaintext combinations $k \in \{0, p-1\}$, $x \in \{0, p-1\}$.

In Figure 2, we plot the evolution of the t-statistic, for different choices of the fixed key, as the number of collected samples increases, stopping our collection at 10 million samples. As we can see, there is no significant difference in the behavior observed across the various plaintext/key combinations. More importantly, the t-statistic is far below both the critical threshold $\bar{t} = 10$ above which the tool reports a function as non-constant time, and the more conservative threshold $\bar{t} = 4.5$ that is more commonly employed for leakage

assessment [SM16].

While extensions of the tool also support execution on ARM devices[1][2], we can obtain a more accurate result directly from power traces collected via the capture board (and inspection of the generated assembly code). Indeed, the numbers provided in Table 1 are cycle-accurate, and do not vary regardless of the input data. This extends to the full encryption algorithm (see Section 5.4).

4 Power Side-channel Analysis of MiMC

Side-channel analysis (SCA) attempts to learn cryptographic secrets by using additional information from physical effects which arise due to implementing an algorithm on real devices. Such physical effects include (but are not limited to) device power consumption, timing variations or electromagnetic radiation. Using additional information leaked from these so-called physical side-channels, an adversary can mount an attack to learn cryptographic secrets. Examples of such attacks are Differential Power Analysis (DPA) [KJJ99], Correlation Power Analysis (CPA) [BCO04], Template Attacks (TA) [CRR02], Soft-analytical SCA (SASCA) [VGS14] and several machine/deep-learning assisted attacks [HGG20, PPM+23]. In these attacks, an adversary collects a number of measurements of the physical side-channel (referred to as *traces*) and then uses statistical analysis with a leakage model to determine the most-likely secret-key (or portion of the key called a "subkey") which generated the traces. The main constraint for the adversary is the number of side-channel traces they are able to collect and thus serves as a proxy for overall attack complexity.

Power analysis attacks can be further divided into two categories: *unprofiled* and *profiled* attacks. Unprofiled attacks generalize the power consumption of a device to follow a generic leakage model e.g., power consumption being proportional to Hamming weights of intermediates. These attacks are very powerful since they make very few assumptions on device leakage behaviors and thus can be applied to a large variety of microcontrollers, processors and dedicated ASICs. Additionally, they can utilize simple statistical analysis methods to determine secret-keys such as Pearson's or Spearman's correlation. Despite their strengths, these attacks typically require a large amount of traces, and masking countermeasures based on secret-sharing [ISW03] have been proven secure against such attacks [PR13a]. In contrast, profiled attacks estimate the true side-channel leakage distribution using probability models or learning techniques. These attacks require an offline profiling step to learn the leakage distribution before carrying out the attack. Despite the learned leakage distribution being specific to a small class of devices, these attacks can enable adversaries to learn secret-keys with as little as a single trace. Furthermore, attacks based on profiled distributions have been shown to be effective against masking countermeasures [BS21]. We investigate both strategies in our analysis of MiMC in the following sections.

4.1 Practical Power Analysis Attacks on MiMC

In this section, we describe two efficient and effective side-channel attack applicable to all variants of MiMC. Concretely, we attack the modular addition in the key-whitening step shown in Line 2 of Algorithm 1 with chosen plaintexts.

Attacking modular addition is difficult due to the weak non-linearity of addition as opposed to the output of an S-box, and due to the carry-bit propagation. Zohner et al. [ZKS12] observed multiple subkey candidates with high likelihood due to these effects in their attack on Skein's modular addition. Specifically, they showed that candidates with small Hamming distances to the correct key tended to have higher correlation (similar to

[1] `https://github.com/dfaranha/dudect-arm`

[2] `https://github.com/itzmeanjan/dudect/tree/make-it-more-portable`

attacks on XORs) and candidates which differed in more significant bits tended to have higher correlation opposed to candidates which differed in less significant bits within the same Hamming distance class. Nevertheless, we opted to attack the key-whitening step for two reasons: (i) the addition allows setting active bytes to enable a divide-and-conquer style attack, and (ii) avoids the high diffusion which occurs as a result of the fifth-order exponentiation in the rest of the round.

Linear regression-based attack. Schindler et al. [SLP05] first described a profiling methodology based on linear regression which was then later refined into a robust side-channel attack by Doget et al. [DPRS11]. Our practical attack on MiMC uses this methodology for attacking the modular addition in the key-whitening step. We found this strategy increased the robustness of the attack and enabled more consistent recovery of correct key-bytes in preliminary tests. This is most likely due to the increased difficulty of attacking modular addition. The increased robustness is due to the assumption that the observed leakages of an n-bit intermediate x are proportional to a linear combination of its bits,

$$m(x) = \alpha_0 + \sum_{j=1}^{n} x\alpha_j, \tag{4}$$

where a portion of the collected traces are used to learn the model function $m(x)$ on the fly. We briefly describe the linear regression attack and refer the reader to the work of Doget et al. [DPRS11] for more details and comparisons.

We attack the target instance of MiMC described in Section 2.1. Let $k \in \mathbb{F}_p$ be the 254-bit secret-key and let k^i represent the i-th byte in the byte string represention of k: $\left(k^{31}, \ldots, k^0\right)$. Similarly, let $x \in \mathbb{F}_p$ be the chosen plaintext with x^i representing the i-th byte in the same byte string representation. To learn byte k^j, the adversary sets all other bytes $i \neq j$ to the zero byte `0x00` in x and x^j to a chosen value. Then, the adversary makes N queries with different chosen values for x^j and records the leakages.

Let $\boldsymbol{\ell}_{k^j}$ represent the N leakage measurements collected by the adversary containing information on the modular addition $v_k^j = x^j + k^j$. The adversary then chooses a key-byte guess $\hat{k}$ for j to form the hypothesis $v_{\hat{k}}^j = \hat{k} + x^j$. Let $\boldsymbol{M}_{\hat{k}}^j$ be an $N \times 10$ matrix containing a column of 1s followed by the 9-bit binary decomposition of the N hypothetical $\left\{v_{\hat{k}}^j\right\}^N$ (an extra bit is included to capture the carry-bit overflow),

$$\boldsymbol{M}_{\hat{k}}^j = \begin{bmatrix} 1 & v_{\hat{k}}^{j,1}[8] & \cdots & v_{\hat{k}}^{j,1}[0] \\ \vdots & \vdots & \ddots & \vdots \\ 1 & v_{\hat{k}}^{j,N}[8] & \cdots & v_{\hat{k}}^{j,N}[0] \end{bmatrix}. \tag{5}$$

The adversary solves for the coefficient vector, $\boldsymbol{\alpha}_{\hat{k}}^j$, in the matrix equation,

$$\boldsymbol{\ell}_{k^j} = \boldsymbol{M}_{\hat{k}}^j \cdot \boldsymbol{\alpha}_{\hat{k}}^j, \tag{6}$$

using the *ordinary least squares* method,

$$\boldsymbol{\alpha}_{\hat{k}}^j = \left(\boldsymbol{M}_{\hat{k}}^{j\intercal} \cdot \boldsymbol{M}_{\hat{k}}^j\right)^{-1} \cdot \boldsymbol{M}_{\hat{k}}^{j\intercal} \cdot \boldsymbol{\ell}_{k^j}. \tag{7}$$

The goal of the adversary is to select the key-guess $\hat{k}$ such that the score function is minimized,

$$s_{\hat{k}}^j = \left\| \boldsymbol{\ell}_{k^j} - \boldsymbol{M}_{\hat{k}}^j \cdot \boldsymbol{\alpha}_{\hat{k}}^j \right\|^2. \tag{8}$$

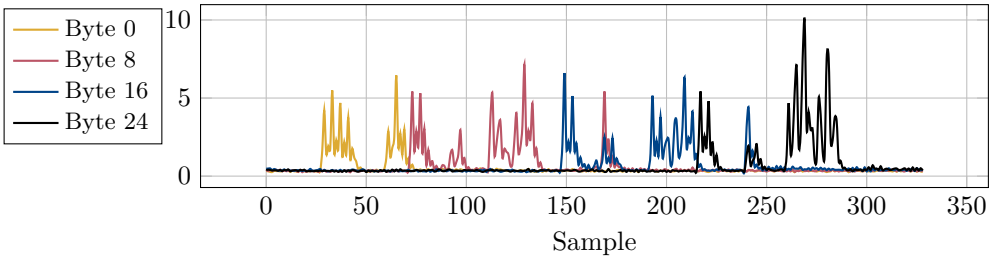


Figure 3: Signal-to-noise ratio of the key-addition for four selected bytes computed from 1000 measurements of a known key-byte and known plaintext-byte.

Additionally, they can compute the probability of observing a key guess given the leakages [CPS16, Eq. 8],

$$\Pr\left[\hat{k} \,\middle|\, \boldsymbol{\ell}_{k^j}\right] = \frac{\mathrm{std}\left(\boldsymbol{\ell}_{k^j} - \boldsymbol{M}^j_{\hat{k}} \cdot \boldsymbol{\alpha}^j_{\hat{k}}\right)^{-N}}{\sum_{k'=0}^{255} \mathrm{std}\left(\boldsymbol{\ell}_{k^j} - \boldsymbol{M}^j_{k'} \cdot \boldsymbol{\alpha}^j_{k'}\right)^{-N}}. \tag{9}$$

For a key-byte k^j, the adversary will observe *two* key-candidates with equal likelihood of being the correct key candidate. This result is exactly due to the correct key-byte and the corresponding byte with a flipped MSB having the same variance for the chosen inputs cf. [ZKS12]. Therefore, a guess must be made between two key-byte candidates for each of the 32 key-bytes (except for the last byte, since the MSB of this byte is known to be zero), requiring the adversary to enumerate $\approx 2^{31}$ possible key-candidates for the full 254-bit secret-key recovery.

Experimental results. To verify our theoretical attack complexity, we implemented the linear-regression attack on a commodity off-the-shelf ARM Cortex-M4 microcontroller running the optimized constant-time implementation described in Section 3. Our experimental setup uses the ChipWhisperer CW308 UFO board [Inc24] with the STM32F303 target. We use the CW1173 ChipWhisperer-Lite capture board to record measurement traces. The target is clocked at 7.37 MHz and the sample frequency is set to 4 times the target clock frequency. We record the entire key-whitening addition operation in the first round.

First, we collected traces with known keys and plaintexts to find points-of-interest. With this dataset, we estimated the univariate signal-to-noise ratio metric with 1000 traces collected per byte. In Figure 3, we depict the SNR measurements for the first four bytes in each 64-bit word. To conduct the attack, we first generate a fixed random 254-bit key from $\mathbb{F}_p$. Then for each byte position, we generate 1000 valid messages with random values with all other bytes set to the zero byte `0x00` and record the power traces. We select a single POI based on the top n highest SNRs to construct the $\boldsymbol{\ell}_{k^j}$ vector for the linear regression attack for each byte individually. In Figure 4, we show the results of the linear regression attack with the log-likelihoods computed from Equation (9). These results show that both the correct key and the corresponding byte with a flipped MSB are equally likely candidates. From these results, we compute the rank estimation for different numbers of traces to understand the overall attack complexity in Figure 5. We use rank estimation algorithm implemented by the SCALib library [CB23] which is based on histogram convolution [PSG16]. The results show we are able to achieve a $\log_2$ key rank of $\approx 2^{31}$ with about 1000 traces per byte, or 32 000 traces in total. Thus, we experimentally verify the theoretical minimum attack complexity given in our analysis in the previous section.

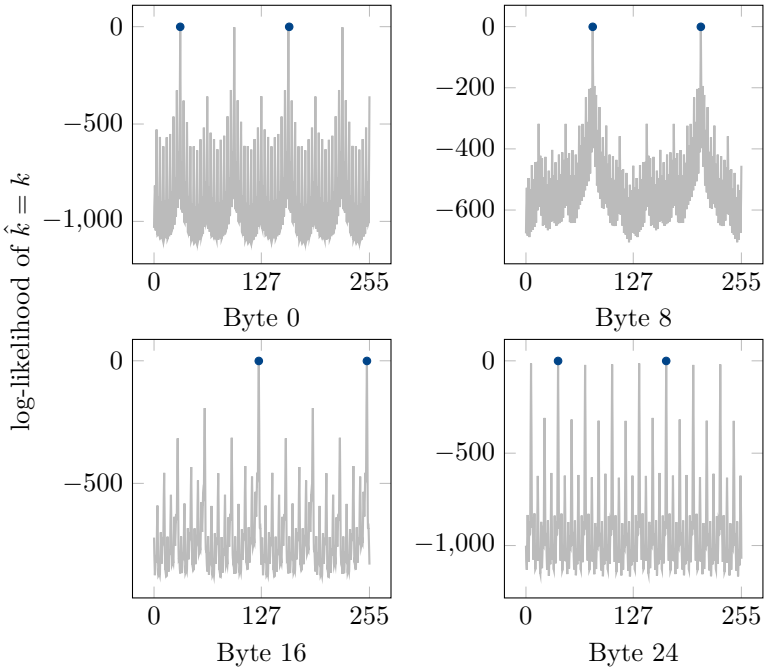


Figure 4: Log-likelihood of observing a key-guess given the leakage vector $\boldsymbol{\ell}_{k^j}$ computed using Equation (9) from $N = 1000$ traces. The • indicates the two guesses with the highest likelihood, corresponding to the right key and its counterpart with a flipped MSB.

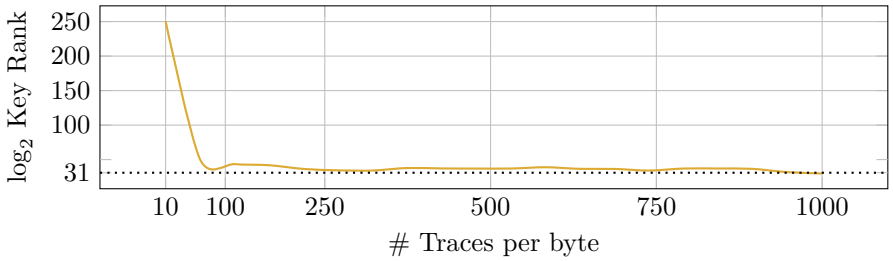


Figure 5: Result of rank estimation for a fixed sized set of traces.

4.2 Profiling Attacks on MiMC

We now analyze MiMC from the perspective of an adversary capable of performing a profiled attack. Concretely, we demonstrate a practical SASCA attack with leakage models constructed from multivariate Gaussian templates in reduced dimensionality spaces [SA08].

Using the notation from the previous section, let $\mathcal{D}_p$ be the profiling dataset of n_p profiling traces the adversary constructs with known keys, $k \in \mathbb{F}_p$ and plaintexts, $x \in \mathbb{F}_p$, of the target intermediate, $v = x + k$: $\mathcal{D}_p = \{(v_i, \boldsymbol{\ell}_i)\}_{i=1}^{n_p}$. Similarly, let $\mathcal{D}_a$ be the attack dataset of n_a attack traces the adversary constructs of the intermediate v with a fixed, but unknown, key $k^* \in \mathbb{F}_p$. The leakage models for each byte in v are constructed using multivariate Gaussian templates projected into a p-dimensional subspace by finding the $p \times n_s$ dimensionality reduction matrix $\boldsymbol{W}$ which maximizes the ratio between the inter- and intra-class scatter matrices (We refer the reader to [SA08] for a more detailed treatment).

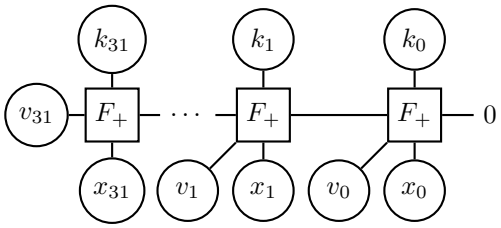


Figure 6: Factor graph used to model the full 32 byte modular addition.

The leakage model is constructed from the profiling dataset for the i-th byte in v with

$$f\left[\boldsymbol{\ell} \mid v^i = (k^{*,i}, x^i)\right] = \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}_i|}} \exp\left(-\frac{1}{2}(\boldsymbol{W}_i\boldsymbol{\ell} - \mu_{v^i})^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{W}_i\boldsymbol{\ell} - \mu_{v^i})\right), \tag{10}$$

where the mean μ_{v^i} and covariance matrix $\boldsymbol{\Sigma}_i$ specify the noise distribution for each byte class in the reduced subspace.

The templating model assumes that each byte in the intermediate v is computed independently, which is not the case in modular addition where the i-th bit depends on the previous $(i-1)$-th carry bit. To capture this dependency, we encode the byte-wise sum-carry relationship into a factor graph and use the belief propagation algorithm of Pearl et al. [Pea82] to compute the exact marginals of each key byte (for more details, see [VGS14, NCP+24]). The sum-carry factor is defined as the indicator function for a full 8-bit adder,

$$F_+(a, b, c_{\text{in}}, s, c_{out}) = \mathbb{1}\left[s = a + b + c_{\text{in}} \pmod{256} \wedge c_{\text{out}} = \left\lfloor \frac{a+b}{256} \right\rfloor\right]. \tag{11}$$

Next in the attack phase, the adversary uses the generated models to construct a score vector $\boldsymbol{S}^i_{n_a}$ for each possible key byte candidate in the key space $k^i \in \mathcal{K}$,

$$\boldsymbol{S}^i_{n_a}(\mathcal{K}) = \left\{ \sum_{j=1}^{n_a} \log_2\left(f\left[\boldsymbol{\ell}^*_j \mid v^i_j = (k^i, x^i_j)\right]\right) \right\}_{k^i \in \mathcal{K}}. \tag{12}$$

If the correct key byte $k^{*,i}$ is ranked within the top n in the sorted $\boldsymbol{S}^i_{n_a}$ score vector, then the attack is considered successful.

Profiling metrics. The stochastic nature of profiling side-channel analysis makes attack performance difficult to characterize and thus necessitates a variety of metrics to evaluate and compare successful attacks. In addition to the key rank, we utilize the *guessing entropy* which we define to be the average rank of the correct key after several independent experiments with n_a traces,

$$\text{GE} = \mathbb{E}\left[\log_2 \text{rank}(k^*, n_a)\right]. \tag{13}$$

To evaluate model quality, we compute the perceived information, the empirical lower bound of the mutual information,

$$\text{PI} = H[\mathcal{K}] + \frac{1}{n}\sum_{j=1}^{n} \log_2\left(\Pr_f\left[v_j \mid \ell_j\right]\right), \tag{14}$$

using a subset of the profiling traces unknown to the model.

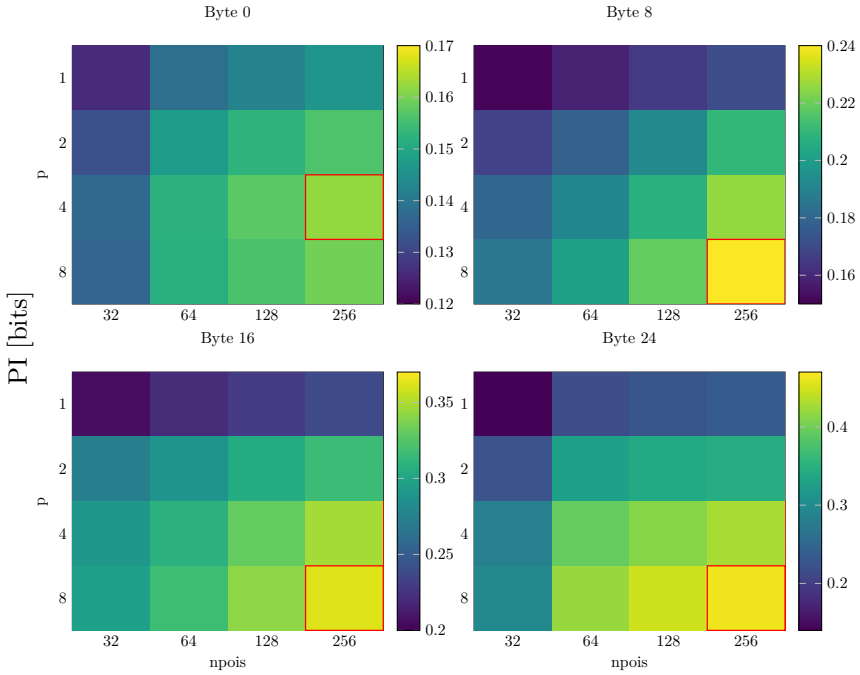


Figure 7: PI-based parameter selection for selected bytes. The parameters with the highest PI are boxed in red.

Experimental results. We carry out the SASCA attack using the same setup for the linear regression attack. The profiling dataset consists of 10^6 traces, and we use a 90-10 split for parameter selection. To construct templates for each byte, we restrict the number of POIs used in template construction to be the top n_s points with the highest SNR within the profiling. Then, we select the best model for each byte by finding the parameter selection of (p, n_s) which maximizes the perceived information. The result of this parameter selection is given in Figure 7.

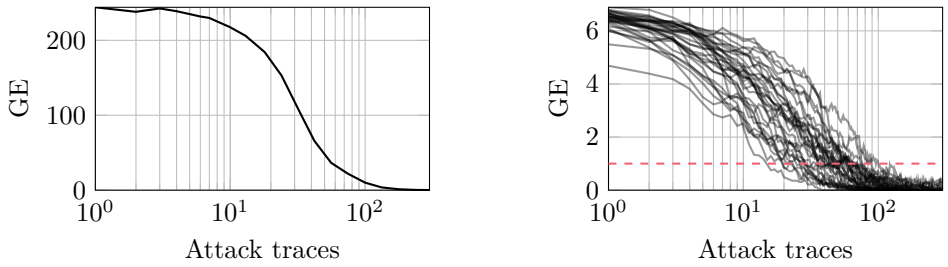
Using the best models for each byte, we performed 25 individual SASCA attacks and report the guessing entropy of the master key in Figure 8. We consider the attack to be successful when $\mathrm{GE} < 1$ since it is more accurate to characterize attack performance as the number of experiments increases [ZL25]. Compared to the linear regression attack, the SASCA attack is able to recover the full 254-bit key in approximately 100 traces.

5 Protecting MiMC from Power Analysis

In Section 4, we demonstrated a practical power analysis attack on the constant-time implementation of MiMC. We will now turn our attention towards two types of countermeasures that can be applied to protect MiMC from such attacks, namely secret-sharing-based masking (or simply masking), and shuffling based on a redundant number representation (RNR).

5.1 Secret-Sharing-Based Masking

Secret-sharing-based *masking schemes*, such as the famous ISW approach [ISW03], have traditionally been one of the most common approaches to protect cryptographic implementations from probing side channel attacks like power analysis.



(a) Full key. (b) Per byte.

Figure 8: Guessing entropy of the key after 25 individual SASCA attacks.

A t-order masking scheme over a field $\mathbb{F}$ consists in a transformation of the representation of the field elements into a sequence of $t+1$ shares, together with a rewriting of the operations of field addition and multiplication that maintains the expected input-output relationship without leaking information about the secret.

For example, in first-order masking, in order to share some value s, we randomly sample a secret share $s_1 \xleftarrow{\$} \mathbb{F}$, and compute the second share as $s_2 = s + s_1$. Then, to unmask the secret, we may simply subtract the first share from the second. Addition of two masked values can be securely implemented via component-wise addition of the respective shares. Indeed, given $(s_1 + s_1', s_2 + s_2')$, we have that $s_2 + s_2' - (s_1 + s_1') = s + s'$. To multiply two masked inputs while ensuring independence of the result, we first need to generate a fresh random value $r \xleftarrow{\$} \mathbb{F}$, and then evaluate the new shares as:

$$\begin{pmatrix} s_1 s_1' + r \\ s_1 s_2' + s_2 s_1' + s_2 s_2' + r \end{pmatrix} \tag{15}$$

Hence, each multiplication, when masked at the first order, requires evaluating four multiplications and one share refresh.

The most popular type of masking schemes is *boolean masking*, where $\mathbb{F} = \mathbb{F}_2$. While any arithmetic circuit can be transformed into an equivalent boolean circuit, changing the representation will result in a vastly larger circuit.

A more suitable alternative is hence *arithmetic masking*, where $\mathbb{F}$ can be any field, and in particular it can be the same prime field over which our MiMC instance is defined. While there has been some work into optimizing masking schemes for arithmetic circuits over fields of even characteristic [CGP+12], only recently arithmetic masking over prime fields has gained traction [MMMS23, CMM+23], after it was shown that prime-field masking schemes can result in the amplification of arbitrarily small noise levels, while this not necessarily the case over composite-order fields [DFS16].

For MiMC, arithmetic masking is clearly the more natural and efficient approach. However, even arithmetic masking introduces a significant overhead: in order to perform a single share-wise multiplication, we now need four field multiplications, while squaring can be optimized to require only one multiplication and two squarings. As plain MiMC requires one multiplication and two squaring operations per round, it follows immediately that for a single first-order masked round we need 6 multiplications and 4 squaring operations, and several sums. Further, we also need to generate 3 random fresh shares per round. Since in order to ensure uniformity of distribution we must employ a rejection sampling algorithm over a random bit generator, and considering that each field element fits into nw bits, we need to generate, on average, about $nw \leq 2^{\lceil \log_2(p) \rceil}/p \leq 2nw$ random bits per refresh. In our target instance, given that $r = 110$ and $nw = 256$, we would need to generate at least $28,160$ random bits of data for a single block encryption. In conclusion, first-order masking incurs in a significant overhead compared to the plain constant-time implementation,

making the resulting algorithm highly expensive to execute (see Section 5.4), heavily depending on the ability of the randomness source to provide many high-entropy random bits at a high rate.

5.2 Shuffled Redundant Number Representation

An alternative approach to masking, that has been successfully applied in the context of lattice-based primitives for post-quantum cryptography [ZBT19, HP23b, NHPM25], is the so-called a redundant number representation (RNR) countermeasure. The core idea lies on the observation that the equivalence classes in $\mathbb{Z}/p\mathbb{Z}$ has infinite representatives, and by executing some arithmetic circuit over a random representative, rather than over the canonical one, we have a natural way of reaching the same result (modulo p) while having a different power profile. Formally:

Definition 2 (Shuffled RNR)**.** Let $\mathbb{F}_p \cong \mathbb{Z}/p\mathbb{Z}$ be the finite field of prime order p, and let $\eta \in \mathbb{N}$ be some arbitrary constant. Then, given the canonical representative $a \in \{0, \ldots, p-1\}$ of a field element $x \in \mathbb{F}_p$, a shuffled RNR-η representation of x is given by $a_s = a + sp$, where $s \xleftarrow{\$} \{0, \ldots, \eta - 1\}$.

We omit specifying η when it is clear from the context.

Since, for performance reasons, we want to be able to do Montgomery arithmetic, we need the value $\tilde{\eta p} = 1/\eta p \mod 2^w$ to be well-defined, which is the case only if ηp, and hence η, is odd.

Proposition 1 (RNR for MiMC-p)**.** *Given a prime number p and a positive integer η, let* MiMC$_\eta$-p *denote the map defined in the same way as* MiMC-p *but with domain and range lifted to $\mathbb{Z}_{\eta p}$. Then, for all $x, k \in \{0, \ldots, p-1\}$, and all $s_x, s_k \in \{0, \ldots, \eta - 1\}$, it holds that:*

$$\text{MiMC}_\eta\text{-}p(k + s_k p, x + s_x p) \bmod p = \text{MiMC-}p(k, x) \tag{16}$$

Proof. To prove equivalence for the lifted construction, we simply need to prove equivalence for the lifted addition and multiplication. It's immediate to see that, for any $p, x, y, s, t \in \mathbb{N}$, it holds that:

$$((x + sp) + (y + tp)) \bmod \eta p \equiv x + y + (s + t)p \equiv x + y \pmod p \tag{17}$$

And similarly:

$$((x + sp) \cdot (y + tp)) \bmod \eta p \equiv xy + (tx + sy)p \equiv xy \pmod p \tag{18}$$

Since congruence modulo p is preserved for addition and multiplication modulo ηp, and those are the only two operations used by MiMC$_\eta$-p, the claim follows. □

5.3 Effectiveness of the Shuffled RNR

Naturally, we expect larger values of η to provide better randomization, this comes at a computational cost. We can gain some insights about the choice of η by considering a 'windowed' Hamming Weight (HW) leakage model. Recall that the HW of a bit-string is the number of set bits in the string. When interpreting bit strings as base-2 unsigned integers, we can define:

$$\forall x \in \mathbb{N}\colon \text{hw}_l^h(x) = \sum_{i=l}^{h} \left\lfloor \frac{x}{2^i} \right\rfloor \bmod 2, \tag{19}$$

and let $\text{hw}(x) = \text{hw}_0^\infty(x)$. For x uniform over $\{0,1\}^n$, then $\text{hw}(x)$ follows a binomial distribution over $\{0, \ldots, n\}$, and hence $\Pr[\text{hw}(x) = b] = \binom{n}{b}/2^n$. Further, since each bit

contributes independently to the total weight, we similarly have that $\Pr\left[\mathrm{hw}_l^h(x) = b\right] = \binom{h-l+1}{b}/2^{h-l+1}$.

Now, given some positive integers η and a, let $[a]_p^\eta = \{a + sp \mid s \in \mathbb{Z}_\eta\}$. We will drop p and η from the notation when clear from the context. Additionally, we will assume $0 \leq a < p$. Hence, for any $x \in [a]$, it holds that $\mathrm{hw}_l^h(x) = \mathrm{hw}_l^h(a + sp)$, and we are interested in studying the distribution of the latter over a random choice of s.

A degenerate case: powers-of-two moduli. As an example, assume for a moment that p is not an odd prime, but rather $p = 2^n$ for some $n \in \mathbb{N}$. We will see that, if this were the case, the RNR approach would not be effective. Indeed:

$$\mathrm{hw}_l^h(a + sp) = \mathrm{hw}_l^h(a + s2^n) = \mathrm{hw}_l^{n-1}(a) + \mathrm{hw}_n^h(s), \tag{20}$$

where the last equivalence holds since $a < p = 2^n$. Now, if an adversary can learn the word-level Hamming weight, that is $\mathrm{hw}_{iw}^{iw+w-1}(a + sp)$, then the full weight of the low bits of a is leaked whenever $n \geq w$!

The case of (prime) odd moduli. When p is prime[3], like in our case, the situation is different. In particular, the map:

$$f_{p,a,i} \colon \mathbb{Z} \to \mathbb{Z}; s \mapsto a + sp \bmod 2^i \tag{21}$$

induces a periodic permutation over $\mathbb{Z}_{2^i}$, for any valid choice of p, a, since $\gcd_{i\in\mathbb{N}}\left(p, 2^i\right) = 1$ by definition of p. When s ranges over $\mathbb{Z}_\eta$, the sequence repeats exactly $\left\lfloor \frac{\eta}{2^i} \right\rfloor$ times, followed by a remainder tail of length $\ell_i = \eta \bmod 2^i$. Hence, when considering a uniformly distributed s, we effectively have that the first i bits of $a + sp$ are *almost* uniformly distributed, except for the remainder tail, whose content depends on the choice of a and p. But if $\eta \equiv 0 \pmod{2^i}$, then $\ell_i = 0$, and now the first i bits are perfectly uniformly distributed. However, we cannot possibly cancel out the tail for all bit indices, since for any $\eta \neq 0$ there must be some least i such that $\eta \not\equiv 0 \pmod{2^j}$ for all $j \geq i$.

From now on, assume $\eta = 2^\beta$, so that the tail length will be $\ell_i = 2^\beta \bmod 2^i$, which is either 0 if $i \leq \beta$ or 2^β if $i > \beta$. In the latter case, for a particular choice of p there may be two values a_0 and a_1 such that the permutations induced by $f_{p,a_0,i}$ and $f_{p,a_1,i}$ over $\mathbb{Z}_{2^i}$ include, in some order, first elements which are $< 2^\beta$ (respectively $\geq 2^\beta$), followed by elements which are all $\geq 2^\beta$ (respectively $< 2^\beta$), inducing maximally unbalanced tails.

An easy case: Mersenne numbers. While understanding how close to being maximally unbalanced the most unbalanced tail can get depends on the concrete choice of p, we may consider primes with a special structure to better understand what may happen. Mersenne numbers are positive integers that can be written as $2^n - 1$ for some n. Mersenne primes are quite rare (the next Mersenne prime after $2^{127} - 1$ is $2^{521} - 1$). If p is a Mersenne number, we can write:

$$f_{p,a,i}(s) = a + sp \bmod 2^i = a + 2^n s - s \bmod 2^i \tag{22}$$

Now, if $i \leq n$, then $a + 2^n s - s \bmod 2^i = a - s \bmod 2^i$ is a simple rotation of $\mathbb{Z}_{2^i}$. If $\beta < n$, then choosing $i = n$ and $a = 0$ will induce a tail consisting of 0 plus all the elements whose nth bit is 1, while $a = (p-1)/2$ will induce a tail consisting of all the elements whose nth bit is 0. Hence, the distributions of the nth bit in the two cases are (almost) disjoint. On the other hand, if $\beta = n$, then there will be no tail for $i \leq n$. Additionally, the permutation induced over $\mathbb{Z}_{2^i}$ by $a + 2^n s - s \bmod 2^i$, for $i > n$, is such that, for any a, the ith bit is

[3]The reasoning can be applied for any odd modulus > 2.

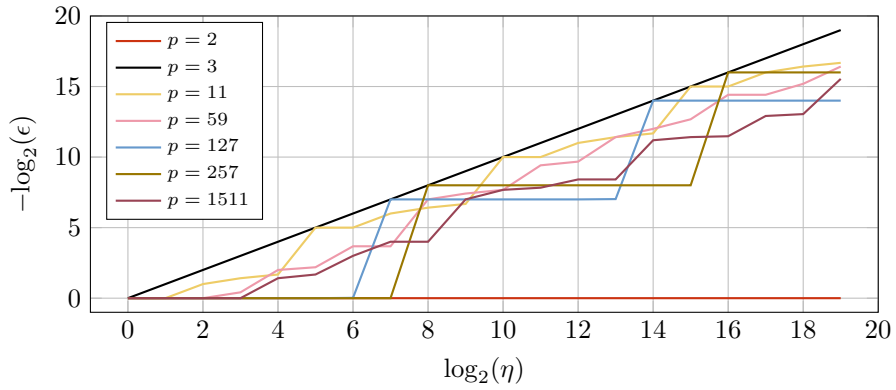


Figure 9: Maximum bit distribution deviation observed across all bits when employing the RNR countermeasure, for increasing values of the redundancy parameter $\eta = 2^\beta$.

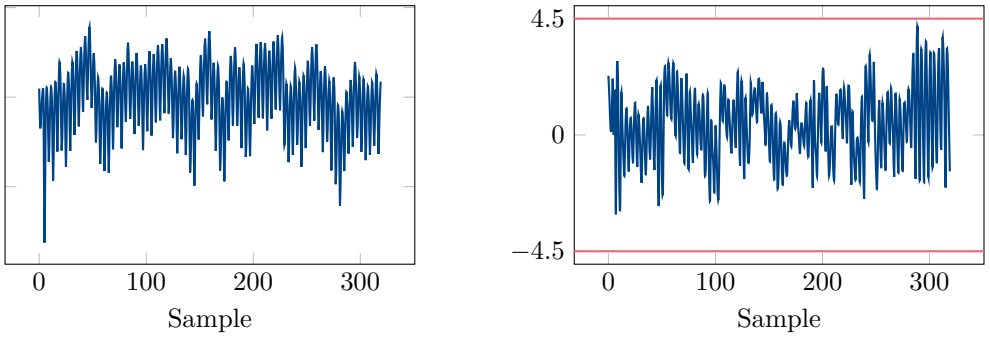
uniformly distributed over the choice of s, except for an extra 0 bit when the parity of the $(i-n)$th bit of a is even. When increasing β beyond n, the deviation doubles for each increment until β is one less than a multiple of n, in which case the deviation is one less than double, and then goes back to only a single-element deviation when β realigns to a multiple of n. However, as we are dealing with increasingly long tails, the relative skew decreases exponentially with β (i.e. the number of redundancy bits), so that the maximum deviation across any two equivalence classes when $\eta = 2^\beta$ is bounded by $\epsilon = \frac{2^{\beta \bmod n}}{2^\beta}$

Generic primes. When considering unstructured primes, working out an exact bound is not straightforward. Experimentally, however, we can still observe an overall consistent trend, as we can see from Figure 9. For example, we observe, as expected, that for $p = 2$ there is no benefit from using RNR, while for $p = 3$ we obtain an optimal, perfectly linear behavior. As another example, $p = 1511$ has better uniformity than $p = 127$ for values of $\eta \in [0, 2^6] \cup [2^9, 2^{13}]$. Indeed, as discussed before Mersenne primes reach optimal distribution uniformity at equidistant choices of $\log_2(\eta)$ as explained before, but there is no improvement in-between these 'jumps'.

Concluding, in general, the maximum deviation from the uniform distribution across equivalence classes provided by the RNR countermeasure decreases roughly linearly with the choice of η (i.e. exponentially in its bit-length), although this does not in general happen 'smoothly'.

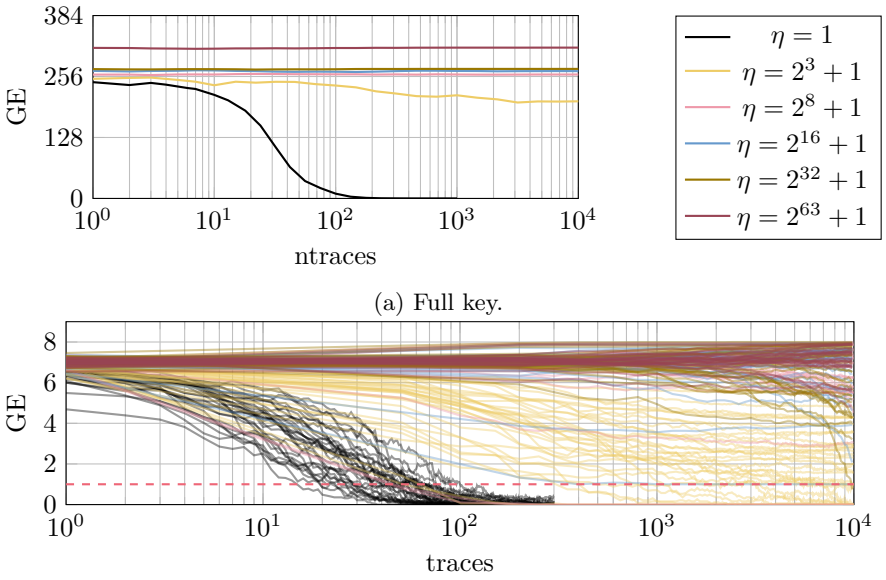
Experimental analysis. In order to establish a suitable choice of η, we employed the well-understood leakage assessment methodology [SM16] based on Welch's t-test. In this test, we record two equally sized sets of traces with one set containing *fixed* inputs and the other containing *random* inputs. We collect measurements using the same attack setup described in Section 4. In Figure 10, we depict the result of the leakage assessment evaluation. The result indicates that the Montgomery-RNR implementation, with η a 64-bit value, does not exhibit data-dependent leakage even after 10 Million traces.

To further support the leakage assessment, we performed SASCAs similar to that in Section 4 for different values of η to better understand how the RNR countermeasure performs against profiling attackers. In this analysis, we sweep over possible $\eta_{\max} \in \{2^n + 1; \forall n = 3, 8, 16, 32, 63\}$ and task the adversary with finding *one* of the RNR-blinded keys, $k \in \mathbb{Z}_{\eta p}$, used in the key-whitening step. During profiling, the adversary is aware of the η values and builds templates using the same strategy in Section 4 for up-to 40 bytes. In the attack phase, the key is fixed, but refreshed with a different η for each invocation.



(a) Power trace. (b) 1st-order t-statistic.

Figure 10: Results of the *fixed vs. random* leakage assessment for the Montgomery-RNR-64 variant using 10 Million traces.



(a) Full key.

(b) Per key byte.

Figure 11: Guessing entropy of the RNR-blinded key for different η averaged over 25 independent SASCA experiments.

The results of the analysis is given in Figure 11. We include the unblinded ($\eta = 1$) case for comparison.

In the unblinded case, the full key can be recovered in as few as 100 traces. However, even for the smallest $\eta = 2^3 + 1$, the total guessing entropy was not reduced below 200 bits even after 10^4 attack traces. Furthermore, the total guessing entropy increases with larger values of η since probability of observing the same blinded key decreases. Thus, against a profiling adversary, RNR significantly increases the difficulty of carrying out a successful attack.

5.4 Masking vs. RNR

When considering which countermeasure to apply, it is important to understand the different guarantees that they can provide in terms of security. To the best of our

Table 2: Performance comparison of constant-time, RNR with a 64-bit η parameter, and ISW first-order masking implementations of MiMC over the BN254 scalar field.

Implementation	cycles/encryption	
	Core i9-13900KF	STM32F4
MiMC-CT	23 133	431 826
MiMC-RNR	33 374	665 362
MiMC-ISW	1 885 052	3 583 285

knowledge, there are no results in the literature which give formal security proofs for the RNR approach, even though the countermeasure itself has been well-known for a long time [SOP08]. Hence, the security of the RNR countermeasure relies mainly on the leakage assessment and experimental evaluation conducted earlier, from which we conclude that a 64-bit value of η is sufficient to make power-based attack impractical. On the other hand, there are many results in the literature that formally prove the security of (boolean) masking [CJRR99, PR13b], although such results do not necessarily carry over to arithmetic masking, and the formal model may diverge from the actual leakage behavior of the device (and of the randomness source). Additionally, while higher-order masking offers better security, it often entails a much higher overhead, so that first-order masking is considered sufficient to thwart practical attacks.

We wrote an implementation for both countermeasures on our x86-64 and ARM Cortex M4 target architectures, and compared them to the baseline constant-time implementation. We stress that our implementations are proof-of-concept, and further optimizations are likely possible. Towards this direction, we observe that, as the RNR countermeasure does not operate over shares, precise control of register allocation and instruction scheduling to avoid leakage induced from cross-share interaction is not as relevant as for masking. In order to gather randomness, in the x86 implementation we exploit the `rdrand` instruction, while for the ARM target, we rely on the hardware random number generator available on the STM32F4 MCU. We point out that the two randomness sources have very different latencies, with the STM32F4 RNG being much faster, in relative clock cycles, than the rdrand instruction on the target CPU. From Table 2, we can see that, on both ARM and x86 targets, the RNR countermeasure introduces approximately a 50% overhead compared to the baseline constant-time implementation. On the other hand, first-order masking incurs into a much higher 8× overhead on the ARM target, and an even higher 80× overhead on the x86 target.

6 Conclusions

In the last decade, Arithmetization-Oriented cryptography has evolved from a niche area into a widely adopted solution to facilitate the deployment of privacy-enhancing protocols such as ZKP, MPC and FHE. As these protocols become more relevant in embedded IoT contexts, it is important to have AO primitives that can be effectively protected against timing and power side-channel attacks. As a first step in this direction, we considered one of the earliest and simplest AO block cipher designs, MiMC. We first described its constant-time implementation, which is nevertheless vulnerable against power-based key-recovery attacks. We then considered two potential countermeasures to protect MiMC against power attacks: ISW-like masking and RNR-based shuffling. While the effectiveness of the masking approach is well-known in the literature, offering in particular provable-security guarantees, we observed the (heuristic) RNR approach to be a more fitting alternative for AO ciphers such as MiMC. In particular, our analysis shows that the RNR countermeasure can render power analysis effectively impractical, at a considerably smaller overhead.

Acknowledgements

We thank the anonymous reviewers for their comments, which have been precious for the final version of this work. Stefano Trevisani and Elena Andreeva were funded in whole or in part by the Austrian Science Fund (FWF), grant number 10.55776/F85. Elena Andreeva and Arnab Roy were funded in part by the Vienna Science and Technology Fund (WWTF) CrossPings, grant number 10.47379/ICT25075.

References

[AABS+19] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Paper 2019/426, 2019.

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for mpc, and more. In *Computer Security – ESORICS 2019*, pages 151–171. Springer International Publishing, 2019.

[AGR+16] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *Advances in Cryptology – ASIACRYPT 2016*, pages 191–219. Springer Berlin Heidelberg, 2016.

[ALNS13] Tolga Acar, Kristin Lauter, Michael Naehrig, and Daniel Shumow. Affine pairings on arm. In *Pairing-Based Cryptography – Pairing 2012*, pages 203–209. Springer Berlin Heidelberg, 2013.

[ASM+24] Anees Ahmed, Nojan Sheybani, Davi Moreno, Nges Brian Njungle, Tengkai Gong, Michel A. Kinsy, and Farinaz Koushanfar. AMAZE: accelerated mimc hardware architecture for zero-knowledge applications on the edge. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2024, Newark Liberty International Airport Marriott, NJ, USA, October 27-31, 2024*, pages 194:1–194:9. ACM, 2024.

[Bar87] Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In *Advances in Cryptology — CRYPTO' 86*, pages 311–323. Springer Berlin Heidelberg, 1987.

[BBLP22] Augustin Bariant, Clémence Bouvier, Gaëtan Leurent, and Léo Perrin. Algebraic attacks against some arithmetization-oriented primitives. *IACR Trans. Symmetric Cryptol.*, pages 73–101, 2022.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 16–29. Springer, 2004.

[BM06] Joseph Bonneau and Ilya Mironov. Cache-collision timing attacks against aes. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 201–215. Springer Berlin Heidelberg, 2006.

[BM17] Joppe W. Bos and Peter L. Montgomery. Montgomery arithmetic from a software perspective. Cryptology ePrint Archive, Paper 2017/1057, 2017.

[Bon19] Xavier Bonnetain. Collisions on feistel-mimc and univariate gmimc. Cryptology ePrint Archive, Paper 2019/951, 2019.

[BPA+23] Amit Singh Bhati, Erik Pohle, Aysajan Abidin, Elena Andreeva, and Bart Preneel. Let's go eevee! a friendly and suitable family of aead modes for iot-to-cloud secure computation. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2546–2560. Association for Computing Machinery, 2023.

[BR17] Vitalik Buterin and Christian Reitwiessner. EIP-197: Precompiled contracts for optimal ate pairing check on the elliptic curve alt_bn128. `https://eips.ethereum.org/EIPS/eip-197`, 2017. Ethereum Improvement Proposal (EIP) #197.

[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.*, pages 202–234, 2021.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, 2011.

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

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michael Quisquater, and Matthieu Rivain. Higher-order masking schemes for s-boxes. In *Fast Software Encryption*, pages 366–384. Springer Berlin Heidelberg, 2012.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology — CRYPTO' 99*, pages 398–412. Springer Berlin Heidelberg, 1999.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437. Springer International Publishing, 2017.

[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 Transactions on Cryptographic Hardware and Embedded Systems*, pages 482–518, 2023.

[CPS16] Marios O. Choudary, Romain Poussier, and François-Xavier Standaert. Score-based vs. probability-based enumeration - A cautionary note. In *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, pages 137–152, 2016.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 13–28. Springer, 2002.

[DFS16] Stefan Dziembowski, Sebastian Faust, and Maciej Skórski. Optimal amplification of noisy leakages. In *Theory of Cryptography*, pages 291–318. Springer Berlin Heidelberg, 2016.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. In *Computer Security – ESORICS 2013*, pages 1–18. Springer Berlin Heidelberg, 2013.

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptogr. Eng.*, 1:123–144, 2011.

[DSSR24] D. Dhinakaran, S. M. Udhaya Sankar, D. Selvaraj, and S. Edwin Raja. Privacy-preserving data in iot-based cloud systems: A comprehensive survey with ai integration, 2024.

[Dwo15] Morris Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions, 2015.

[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. Cryptology ePrint Archive, Paper 2020/182, 2020.

[FBJ+24] Marcel Fourné, Daniel De Almeida Braga, Jan Jancar, Mohamed Sabt, Peter Schwabe, Gilles Barthe, Pierre-Alain Fouque, and Yasemin Acar. "these results must be false": A usability evaluation of constant-time analysis tools. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 6705–6722. USENIX Association, 2024.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pages 169–178. Association for Computing Machinery, 2009.

[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 21)*, pages 519–535. USENIX Association, 2021.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the poseidon hash function. Cryptology ePrint Archive, Paper 2023/323, 2023.

[GLR+19] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a generalization of substitution-permutation networks: The hades design strategy. Cryptology ePrint Archive, Paper 2019/1107, 2019.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326. Springer Berlin Heidelberg, 2016.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019.

[HBHW22] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. ZCash Improvement Proposals Website, 2022.

[HGG20] Benjamin Hettwer, Stefan Gehrer, and Tim Guneysu. Applications of machine learning techniques in side-channel attacks a survey. *Journal of Cryptographic Engineering*, 10:135–162, 2020.

[HP23a] Daniel Heinz and Thomas Pöppelmann. Combined fault and DPA protection for lattice-based cryptography. *IEEE Trans. Computers*, 72:1055–1066, 2023.

[HP23b] Daniel Heinz and Thomas Pöppelmann. Combined fault and dpa protection for lattice-based cryptography. *IEEE Transactions on Computers*, 72:1055–1066, 2023.

[Inc24] NewAE Technology Inc. Cw308 ufo target board, 2024.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003*, pages 463–481. Springer Berlin Heidelberg, 2003.

[JK97] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In *Fast Software Encryption*, pages 28–40. Springer Berlin Heidelberg, 1997.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397. Springer, 1999.

[KKAK96] C. Kaya Koc, T. Acar, and B.S. Kaliski. Analyzing and comparing montgomery multiplication algorithms. *IEEE Micro*, 16:26–33, 1996.

[Knu98] Donald Ervin Knuth. *The art of computer programming, Volume II: Seminumerical Algorithms, 3rd Edition*. Addison-Wesley, 1998.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 830–842. Association for Computing Machinery, 2016.

[LP19] Chaoyun Li and Bart Preneel. Improved interpolation attacks on cryptographic primitives of low algebraic degree. In *Selected Areas in Cryptography - SAC 2019: 26th International Conference, Waterloo, ON, Canada, August 12-16, 2019, Revised Selected Papers*, pages 171–193. Springer-Verlag, 2019.

[MMMS23] Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert. Effective and efficient masking with low noise using small-mersenne-prime ciphers. In *Advances in Cryptology – EUROCRYPT 2023*, pages 596–627. Springer Nature Switzerland, 2023.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

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

[Moz] Mozaik project.

[NCP+24] Rishub Nagpal, Gaëtan Cassiers, Robert Primas, Christian Knoll, Franz Pernkopf, and Stefan Mangard. On loopy belief propagation for sascas: An analysis and empirical study of the inference problem. *IACR Commun. Cryptol.*, 1:15, 2024.

[NHPM25] Rishub Nagpal, Vedad Hadžić, Robert Primas, and Stefan Mangard. Efficient SPA countermeasures using redundant number representation with application to ML-KEM. Cryptology ePrint Archive, Paper 2025/679, 2025.

[NK95] Kaisa Nyberg and Lars Ramkilde Knudsen. Provable security against a differential attack. *Journal of Cryptology*, 8:27–37, 1995.

[Pea82] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, USA, August 18-20, 1982*, pages 133–136. AAAI Press, 1982.

[PGS23] Rohini Poolat Parameswarath, Prosanta Gope, and Biplab Sikdar. A privacy-preserving authenticated key exchange protocol for v2g communications using ssi. *IEEE Transactions on Vehicular Technology*, 72:14771–14786, 2023.

[PPM+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55:227:1–227:35, 2023.

[PR13a] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 142–159. Springer, 2013.

[PR13b] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 142–159, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 61–81. Springer, 2016.

[QWA+19] Sana Qabil, Urooj Waheed, Samia Masood Awan, Yusra Mansoor, and Muhammad Ahsan Khan. A survey on emerging integration of cloud computing and internet of things. In *2019 International Conference on Information Science and Communication Technology (ICISCT)*, pages 1–7, 2019.

[RAS21] Arnab Roy, Elena Andreeva, and Jan Ferdinand Sauer. Interpolation cryptanalysis of unbalanced feistel networks with low degree round functions. In *Selected Areas in Cryptography*, pages 273–300. Springer International Publishing, 2021.

[RBV17] Oscar Reparaz, Josep Balasch, and Ingrid Verbauwhede. Dude, is my code constant time? In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 1701–1706. European Design and Automation Association, 2017.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 84–93. Association for Computing Machinery, 2005.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 411–425. Springer, 2008.

[SD21] Xavier Salleras and Vanesa Daza. Zpie: Zero-knowledge proofs in embedded systems. *Mathematics*, 9, 2021.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, pages 30–46. Springer, 2005.

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

[SOP08] N.P. Smart, E. Oswald, and D. Page. Randomised representations. *IET Information Security*, 2:19–27, 2008.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 282–296. Springer, 2014.

[Wan22] Jonathan Wang. Bn254 for the rest of us. `https://hackmd.io/@jpw/bn254`, 2022.

[WEL47] B. L. WELCH. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34:28–35, 1947.

[ZBT19] Timo Zijlstra, Karim Bigou, and Arnaud Tisserand. Fpga implementation and comparison of protections against scas for rlwe. In *Progress in Cryptology – INDOCRYPT 2019*, pages 535–555. Springer International Publishing, 2019.

[ZKS12] Michael Zohner, Michael Kasper, and Marc Stöttinger. Butterfly-attack on skein's modular addition. In *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, pages 215–230. Springer, 2012.

[ZL25] Junfan Zhu and Jiqiang Lu. Leading degree: A metric for model performance evaluation and hyperparameter tuning in deep learning-based side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 333–361, 2025.