Feistel Structures for MPC, and more

Martin R. Albrecht[1], Lorenzo Grassi[2,3], Léo Perrin[4], Sebastian Ramacher[2], Christian Rechberger[2], Dragos Rotaru[5,6], Arnab Roy[5], and Markus Schofnegger[2] ⋆

[1] Royal Holloway, University of London, UK
[2] IAIK, Graz University of Technology, Austria
[3] Know-Center GmbH, Graz, Austria
[4] Inria, Paris, France
[5] University of Bristol, Bristol, UK
[6] imec-Cosic, Dept. Electrical Engineering, KU Leuven, Belgium

Abstract. Efficient PRP/PRFs are instrumental to the design of cryptographic protocols. We investigate the design of dedicated PRP/PRFs for three application areas - secure multiparty computation (MPC), ZK-SNARK and zero-knowledge (ZK) based PQ signature schemes. In particular, we explore a family of PRFs which are generalizations of the well-known Feistel design approach followed in a previously proposed application specific design - MiMC. Attributing to this approach we call our family of PRP/PRFs GMiMC.

In MPC applications, our construction shows improvements (over MiMC) in throughput by a factor of more than 4 and simultaneously a 5-fold reduction of preprocessing effort, albeit at the cost of a higher latency. Another use-case where MiMC outperforms other designs, in SNARK applications, our design GMiMCHash shows moderate improvement. Additionally, in this case our design benefits from the flexibility of using smaller (prime) fields. In the area of recently proposed ZK-based PQ signature schemes where MiMC was not competitive at all, our new design has 30 times smaller signature size than MiMC.

1 Introduction

Computing on Encrypted Data. Due to an increasing maturity of secure multi-party computation, there are a couple of companies such as Partisia [48],

⋆ L. Grassi has been partialy supported by EU H2020 project Safe-DEED, grant agreement n°825225. S. Ramacher has been partially supported by the Austrian Research Promotion Agency (FFG) within the ICT of the future grants program, grant agreement n°863129 (project IoT4CPS), of the Federal Ministry for Transport, Innovation and Technology (BMVIT) and by A-SIT Secure Information Technology Center Austria. D. Rotaru was supported by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. N66001-15-C-4070. Arnab Roy is supported by the EPSRC funding under grant No. EPSRC EP/N011635/1

Sepior [51], Sharemind [16], Unbound [56] which try to incorporate MPC frameworks into large projects to offer services where the companies do not need to know the user inputs to be able to compute on them [10]. Since the complexity of these systems grows, one must be able to incorporate encrypted databases with an MPC system to deal with data in transit or at rest.

For example, the trivial way of storing outputs to be later used is for each party in the MPC engine to encrypt their share using a (different) symmetric key and post it to the database. Later on, when the parties have decided to carry further computations on these shares they simply decrypt the ciphertexts using their corresponding keys. Notice that for a given shared secret there are N ciphertexts where N is the number of parties. This is where cryptographic primitives such as block ciphers play an important role in storing outputs from the MPC engine into an encrypted database: parties can engage in an MPC protocol to compute an encryption of the share using a shared key. In this way, parties jointly produce a single ciphertext rather than having N ciphertexts per stored share.

If one chooses AES as the underlying primitive for the encryption scheme then the share conversions become the bottleneck of MPC procedures when the underlying engine performs arithmetic modulo p. This is indeed the case for most of the frameworks such as MP-SPDZ [7], SCALE-MAMBA [6], BDOZa [13], VIFF [27] and conversion to their boolean counterpart with same security properties is an expensive task. Hence, for efficient and secure computation of algorithms modulo p we would like a blockcipher over the same field. Grassi et al. [35] give several constructions for lightweight pseudorandom functions (PRFs) when evaluated in a *prime field* of large characteristic and concluded that among various other options MiMC [4] is competitive, which is the starting point of our design as well.

Besides database storage, MPC-friendly PRFs can cover other use-cases as well explored in [18, 35]. These include searchable encryption, authenticated encryption, oblivious RAM done in a distributed fashion using MPC and an efficient PRF.

(ZK)SNARK. The most well-known use of (ZK)SNARK is in the area of privacy/anonymity providing cryptocurrency. Zcash [11] is the most popular cryptocurrency which uses this protocol. Zcoin, Zencash are examples of cryptocurrencies based on the (ZK)SNARK protocol. One of the performance bottle-neck in these applications is the lack of "efficient" hash function over suitable (prime) field. In cryptocurrency protocols such hash functions are typically used to insert the (hashed) coin values in a Merkle hash tree. In [4] it was shown for the first time that a hash function designed over prime field are signifincantly faster than SHA2 or SHA3 in SNARK setting. This almost directly speeds up the performance of (ZK)SNARK-based protocols which use SHA2.

ZK-based Signature. Finally, we consider signature schemes based on zero-knowledge proofs of a preimage of a one-way function. It was recently shown that such schemes can be viable alternatives [22, 23] when instantiated with symmetric primitives (to construct a one-way function) that have a low number

of multiplications. Public and private keys are minimized and only consist of a plaintext-ciphertext pair and the secret key, respectively. On top of the post-quantum security of the zero-knowledge proof system (see [21, 32] for recent improvenets of its security analysis), the only hardness assumption required to prove security is the one-wayness of the underlying function. Signature sizes strongly depend on the product of the number of multiplications of the OWF and the size of the field in which the multiplications are performed. The signature and verification times depend on the details of the scheme in a less straightforward way. The block size of the instantiations we are interested in is around 256 bits.

Our Results. In this article, we continue exploring the construction strategies of symmetric cryptography which benefit MPC, (ZK)SNARK and PQ signature applications. We generalize the design approach used in MiMC [4]. More specifically, we use the unpopular design strategy (in symmetric cryptography) – unbalanced Feistel network, and a new balanced Feistel network, for constructing a new family of block ciphers – GMiMC (in Section 2.1) and use it to construct the hash function GMiMCHash (in Section 2.2).
We show the performance of GMiMC in MPC applications based on secret sharing techniques such as BDOZa [13], SPDZ [28] or VIFF [27]. Previous works [35, 49] did not take into account how to optimize the number of multiplications per encrypted share and treated the PRF as a black-box when extending to more inputs. We show that using our construction one can choose to encrypt multiple shares at once thus amortizing the number of multiplications per share and results in a more efficient preprocessing phase. We consider our work to be beneficial when there is a large number of blocks to encrypt. From a theoretical point of view two of our constructions are the first to avoid the linear increase of time and data sent across the parties in the preprocessing phase with the number of encrypted blocks (in $\mathbb{F}_p$). Namely, the cost per encrypted share if we encrypt more shares in one go. Details can be found in Section 6.1.
For (ZK)SNARK applications our design GMiMCHash provides the flexibility of using prime field for smaller primes. For example, GMiMC can be used to obtain a permutation of input size $\approx$ 1024-bit over 128 bit prime field. In MiMC, this permutation could only be constructed using 1024 or 512-bit primes. Additionally, GMiMCHash shows moedrate improvement in performance (see Section 6.2) compared to MiMC.
In the case of the PQ signature scheme, LowMC [2,5] was considered to be clearly the best choice for small signatures and runtimes. MiMC resulted in 10 times larger and hence unpractical signature sizes. As we have shown in Section 6.3, GMiMC is competitive with LowMC and far more efficient than MiMC. This performance is due the flexibility of GMiMC in providing many different field sizes by choosing different branch numbers.

Related Work. Recently, Agrawal et al. [1] considered the problem of parties jointly computing a symmetric-key encryption using a distributed PRF with implications to systems dealing with secret management [47] or enterprise network authentication. Our approach is slightly different since it evaluates the

block-cipher inside the MPC engine. Our result is useful when clients can use it as a standalone tool to encrypt data on their own to then make the encryption compatible with the MPC storage as well.
Secure cryptographic primitives that require a low number of multiplications have many applications. These primitives can reduce the cost of countermeasures against various side-channel attacks [26,36], eliminate the ciphertext expansion in homomorphic encryption schemes [5,20,34,44,45], help dealing with encrypted data in secure multi-party computation systems [5,35,49], increase throughput or latency in SNARKs [4], and reduce the signature size of signature schemes and (privacy-preserving) variants, e.g. ring and EPID group signature schemes, based on one-way functions from symmetric-key primitives and non-interactive zero-knowledge proofs [17,22,29,30,38]. Research efforts in this area are manifold and cover questions on finding circuits for concrete mappings such as S-Boxes [19], foundational theoretical results on the complexity of PRGs, PRFs, and cryptographic hashes [8,9], and new ad-hoc designs of permutations, ciphers and hash functions tailored for various multiplication-related metrics [4,5,20,44].

2 Description of Generalized MiMC

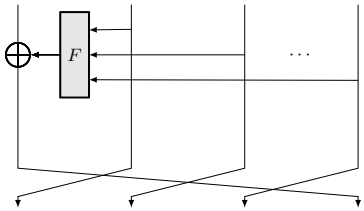
Notation. In a Feistel network, X_{i-1} denotes the input to the branch i, where $1 \leq i \leq t$. X_{t-1} and X_0 denote the inputs to the leftmost and rightmost branches respectively. $X_i \in \mathbb{F}$ for a finite field $\mathbb{F}$. The block size (in bits) of the keyed Feistel permutation is denoted by N, while $n = \lceil \log_2 |\mathbb{F}| \rceil$ denotes the branch size (in bits). We write $\mathbb{F}_p$ for the finite prime field of order p. We write $\mathbb{F}_{2^m}$ for any finite field of order 2^m. The length of the key of a block cipher is given in number of bits and is denoted by κ. In particular, through the paper we work with two different cases (depending on the practical implementation), denoted as:

- the *univariate* case, for which the key-size is $\kappa = n = \lceil \log_2 |\mathbb{F}| \rceil$;
- the *multivariate* case, for which the key-size is $\kappa = N = n \cdot t = \lceil \log_2 |\mathbb{F}| \rceil \cdot t$.

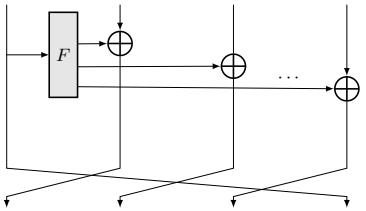
2.1 The Block Cipher GMiMC

We construct "*Generalized Feistel MiMC*" (GMiMC) variants from three generalized Feistel networks, e.g., with contracting round function (CRF), expanding round function (ERF), which are unbalanced Feistel networks and a new balanced Fiestel network which we call Multi-Rotating (MR). Each of the following constructions is a keyed permutation over $\mathbb{F}_{2^n}$ or $\mathbb{F}_p$. The three main parameters of the block ciphers are denoted by $[\kappa, t, n]$. For example, $\text{GMiMC}_{\mathsf{crf}}[4n, 4, n]$ denotes the permutation GMiMC with CRF which has branch size n, key size $4n$ and number of branches 4. The descriptions of each block cipher over $\mathbb{F}_p$ are obtained by replacing the XOR-sum $\oplus$ with the corresponding sum + modulo p. We recall that S-Box$(x) = x^3$ is a permutation in $GF(2^n)$ iff n is odd, while it is a permutation in $GF(p)$ iff $p \neq 1 \mod 3$ (see "Hermite's criterion" for more details).

Fig. 1: One round UFN with CRF and ERF



(a) UFN with CRF



(b) UFN with ERF

GMiMC$_{\mathsf{crf}}$ An unbalanced Feistel network (UFN) with a contracting round function (CRF) can be written as

$$(X_{t-1}, \ldots, X_0) \leftarrow (X_{t-2}, \ldots, X_0, X_{t-1} \oplus F(X_{t-2}, \ldots, X_0))$$

where X_i is the input to the i-th branch of the Feistel network and $F(\cdot)$ is a key-dependent function in round j, cf. Figure 1a. In GMiMC$_{\mathsf{crf}}$ we define the j-th round function as

$$F(x_{t-2}, \ldots, x_0) \coloneqq \Big(\bigoplus_i x_i \oplus k_j \oplus c_j\Big)^3$$

where c_j and k_j are respectively the round constant and the key of the round j (for $1 \leq j \leq r$).

GMiMC$_{\mathsf{erf}}$ An unbalanced Feistel network with an expanding round function (ERF) can be written as

$$(X_{t-1}, \ldots, X_0) \leftarrow (X_{t-2} \oplus F(X_{t-1}), \ldots, X_0 \oplus F(X_{t-1}), X_{t-1})$$

where X_i is the input to the i-th branch of the Feistel network and $F(\cdot)$ is a key-dependent function in round j, cf. Figure 1b. In GMiMC$_{\mathsf{erf}}$ the j-th round function is defined as

$$F(x) \coloneqq (x \oplus k_j \oplus c_j)^3$$

where k_j and c_j are as in GMiMC$_{\mathsf{crf}}$.

GMiMC$_{\mathsf{mrf}}$ In [54], Suzaki and Minematsu introduced new variants of the GFN structure where the linear mixing applied after the Feistel functions is a complex permutation rather than a simple rotation. This allowed them to build GFNs operating on $t = 2^b$ branches such that full diffusion is achieved in $2b$ rounds rather than the $2t$ rounds needed by a Nyberg-style construction [46]. They later used this approach to build the lightweight block cipher TWINE [55].

Here, we introduce the Multi-Rotating structure for generalized Feistel networks, which provides full diffusion as quickly as a TWINE-like structure without

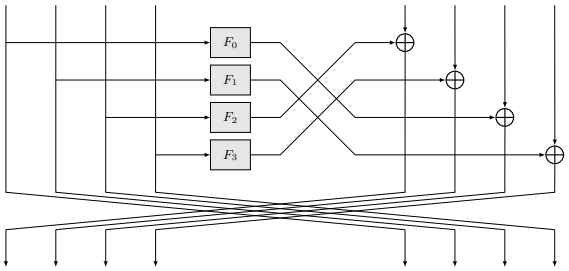


Fig. 2: One round $\mathcal{R}_s$ of an 8-branch Multi-Rotating Feistel network with a rotation by $s = 2$.

the constraint that the number of branches is a power of 2. It is also conceptually much simpler and thus easier in practice to apply to a larger number of branches. These improvements come at the cost of the use of a *different mixing layer in each round* which, to the best of our knowledge, has not been considered for a Feistel or generalized Feistel structure so far. However, we note that previously, the Serpent designers (of SPN type) considered using different mixing layers in each round in their design. In particular, they considered using a different linear transformation for even and odd rounds (see [15, App. A.6]), before settling for their current design.

To introduce the Multi-Rotating Feistel network structure, we first give a general expression of its round function. A *rotated Feistel round* is a permutation $\mathcal{R}_s$ parameterized by a rotation amount s which operates on an even number of branches and works as follows:

$$\begin{aligned}\big(X_{t-1}, \ldots, X_0\big) \leftarrow \big(&X_{t/2-1} + F_{-s}(X_{t-1-g(s,0)}), \ldots, \\ &X_0 \oplus F_{t/2-1-s}(X_{t-1-g(s,t/2-1)}), X_{t-1}, \ldots, X_{t/2}\big),\end{aligned}$$

where the index of the function $F(\cdot)$ is taken modulo $t/2$ and $g(s, i) = s + i \mod t/2$. This process is summarized in Figure 2. Like in GMiMC$_{\mathsf{Nyb}}$, each F_i in the j-th round of GMiMC$_{\mathsf{mrf}}$ is defined as

$$F_i(x) := \big(x \oplus k_{i+j\cdot t/2} \oplus c_{i+j\cdot t/2}\big)^3,$$

where $c_{i+j\cdot t/2}$ are distinct constants in round j and $k_{i+j\cdot t/2}$ are round keys. By iterating such rounds $\mathcal{R}_s$ for varying values of s we obtain a block cipher. An instance of such an r-round block cipher is specified using the sequence $\{s_0, ..., s_{r-1}\}$ of the r rotation amounts used. As explained in [3, App. B], it is possible to build a GFN with optimal diffusion by choosing the sequence $\{s_j\}_{j<r}$ carefully.

We build a GMiMC$_{\mathsf{mrf}}$ instance operating on t branches using a sequence of rotations $\{s_j\}_{0\le j<r}$ where

$$s_{2\ell} = 0 \quad \text{and} \quad s_{2\ell+1} = 2^{\ell \pmod{\lceil \log_2(t/2) \rceil}}. \tag{1}$$

For instance, if $t = 32$, then $\log_2(t/2) = 4$ and this sequence can be written as $\{0, 1, 0, 2, 0, 4,\ 0, 8, 0, 1, 0, 2, 0, 4, 0, 8, 0, 1, ...\}$, i.e. it consists in as many repetitions as needed of the pattern $\{0, 1, 0, 2, 0, 4, 0, 8\}$ of length $2\log_2(t/2) = 8$.

To better understand the security of GMiMC$_{\mathsf{mrf}}$, we now investigate its *diffusion*. We borrow our definition of diffusion from [54]: if a variable y intervenes in the expression of an internal state word X then we say that X *depends* on y. If all output words of a (round-reduced) block cipher depend on all input words, we say that this primitive provides *full diffusion*.

The diffusion provided by GMiMC$_{\mathsf{mrf}}$ using the sequence of rotations from Equation (1) is quantified by the following Theorem, proved in [3, App. B].

Theorem 1. *Let X_j^i denote the word with index j at the input of round i, so that for example X_j^0 denotes a plaintext word. Consider a* GMiMC$_{\mathsf{mrf}}$ *instance operating on t branches with the rotation sequence in Equation 1. If $i \geq 2\lceil\log_2(t)\rceil$, then X_j^i depends on $X_{j'}^0$ for any j, j'. The same is true in the backwards direction. In other words,* GMiMC$_{\mathsf{mrf}}$ *provides full diffusion after $2\lceil\log_2(t)\rceil$ rounds.*

Key Schedule When $|k| = n$ (i.e. the univariate case), then $k_i = k\ \forall i$. The key schedule for the multivariate case $|k| = t \times n$ is a little more complicated. Let $k = k_0||k_1||\ldots||k_{t-1}$, and let M be an *invertible* $t \times t$ matrix with elements in $\mathbb{F}_{2^n}$ or $\mathbb{F}_p$ that satisfies the following condition:

- $\forall i : 1 \leq i \leq \lceil R/t \rceil$ where R is the number of rounds:[7]

$$M^i[j,l] \equiv (\underbrace{M \times M \times .... \times M}_{i\text{-th times}})[j,l] \neq 0$$

  for all $0 \leq j, l < t$, where $X[j,l]$ denotes the coefficient in row j and column l of the matrix X.

For each $1 \leq i \leq \lceil R/t \rceil$ let

$$[k_{i\cdot t}||k_{i\cdot t+1}||\ldots||k_{(i+1)\cdot t-2}||k_{(i+1)\cdot t-1}]^T = M \times [k_{(i-1)\cdot t}||k_{(i-1)\cdot t+1}||\ldots||k_{i\cdot t-2}||k_{i\cdot t-1}]^T.$$

The second condition on M guarantees that *each subkey depends linearly on all the first t subkeys*. This fact has an important consequence. Consider GMiMC$_{\mathsf{crf}}$ and/or GMiMC$_{\mathsf{erf}}$ instantiated with a key schedule that uses the subkeys cyclically, i.e. $k_{i,j} = \hat{k}_{j\cdot t/2+i \pmod t}$. If the attacker guesses $t-1$ subkeys, then she can *potentially* skip both the first and the last $t-1$ rounds. Instead, in the case in which each subkey depends linearly on all the first t subkeys, this strategy simply does not apply. As a result, the proposed key schedule allows to save a certain number of rounds (approximately $t-1$) w.r.t. a key schedule that uses the subkeys cyclically. Similar argumentation holds for GMiMC$_{\mathsf{mrf}}$.

[7] If no matrix exists that satisfies such condition, choose a matrix M for which the total number of zero coefficients for each M^i is minimum.

Round Constants and Number of Rounds For all the above constructions the round constants are generated randomly over the suitable field and fixed. The number of rounds for each of the above block ciphers is chosen to thwart the cryptanalytic attacks mentioned in Section 4. In this article we only provide the number of rounds for the variants which are used in the target applications in Section 6. For the generic formulae of the number of rounds (depending on t, n or p) we refer to the [3].

2.2 Hash Function

To construct the hash function GMiMCHash (over $\mathbb{F}_p$), we use one of the structures, e.g. the $\text{GMiMC}_{\mathsf{erf}}$, with fixed (arbitrary) subkeys.[8] Denoting the fixed key permutation as $\text{GMiMC}^{\pi}_{\mathsf{erf}}[\kappa, t, n]$, GMiMCHash is constructed by instantiating a sponge construction [14] with $\text{GMiMC}^{\pi}_{\mathsf{erf}}[\kappa, t, n]$. The number of rounds of the permutation GMiMC is chosen according to the *univariate* case $2^{\kappa} = 2^n \approx p$.

When the internal permutation $\mathcal{P}$ of an N-bit sponge function (composed of c-bit capacity and r-bit bitrate – $N = c + r$) is modeled as a randomly chosen permutation, it has been proven by Bertoni et al. [14] to be indifferentiable from a random oracle up to $2^{c/2}$ calls to $\mathcal{P}$. In other words, a sponge with a capacity of c provides $2^{c/2}$ collision and $2^{c/2}$ (second) preimage resistance. Given a permutation of size N and a desired security level s, we can hash $r = N - 2s$ bits per call to the permutation.

As usual, the message is first padded according to the sponge specification so that the number of message blocks is a multiple of r, where r is the rate in sponge mode. For GMiMCHash-l we use a GMiMC permutation where $N = n \cdot t = 4 \cdot l + 1$ and $s = 2 \cdot l$. For GMiMCHash-256 we thus use a GMiMC permutation with $N = n \cdot t = 1024$ or 1025. The rate and the capacity are chosen as 512 and 513 respectively. This choice allows for processing the same amount of input bits as SHA-256 (512 bits) while at the same time offering collision security nd preimage security of 256 bits. We highlight that while we could use any of the GMiMC constructions, $\text{GMiMC}_{\mathsf{erf}}$ turns out to be the most efficient choice in several settings as shown in Section 6.2.

3 Security Analysis

We have performed an in-depth security analysis of the GMiMC family of block ciphers (and hash function). In this article we only provide ideas of the most important attacks (in Section 4) which are decisive to the design. Due to the page constraint we refer to the extended version [3] of this article for the details of the analyses.

[8] We emphasize that no key schedule is required in this case, since there is no secret-key material.

Important Remark *Due to our target applications, here we limit ourselves to provide the number of rounds to guarantee security* ***only*** *in the following two scenarios:*

- *GMiMC instantiated over* $\mathbb{F}_p$ with prime size 128 or more (used in SNARKs and MPC applications);
- *GMiMC instantiated over* $\mathbb{F}_{2^n}$ *in the low-data scenario* (used for application like PQ-Signature Scheme).

We stress that this choice is motivated by the fact that we focus only on the scenarios that are useful for our applications.

For some applications like PQ-Signature scheme, the attacker has a limited access to data (e.g. 1 or 2 (plaintext, ciphertext) pairs) as a result, only few attacks (e.g. the GCD one) apply to this case. The security analysis for this particular case is over $\mathbb{F}_{2^n}$ and proposed in [3, Sect. 5]. As the attacker can have access to few (plaintext, ciphertext) pairs, only few attacks (e.g. the GCD one) apply to this case.

Analysis of GMiMCHash over $\mathbb{F}_p$ For the hash function GMiMCHash case, the number of rounds of the inner permutation is chosen according to the corresponding univariate case. This is due to the following considerations. First, as we just recalled in the previous section, when the internal permutation $\mathcal{P}$ of an $N = c+r$ bit sponge function is modeled as a randomly chosen permutation, the sponge hash function is indifferentiable from a random oracle up to $2^{c/2}$ calls to $\mathcal{P}$. The numbers of rounds of the univariate case is sufficient to guarantee security against any (secret-/known-/chosen-) distinguisher which is independent of the key. Equivalently, this means that such number of rounds guarantee that $\mathcal{P}$ does not present any non-random/structural property (among the ones known in the literature). It follows that the previous assumption is satisfied. These and the fact that every key-recovery attack is meaningless in the hash scenario support our choice to consider the univariate case in order to determine the number of rounds of the inner permutation.

4 Security Analysis of GMiMC (over $\mathbb{F}_p$)

Almost all the attacks are independent of the fact that (a) the size of the key is equal to the branch size $\kappa = n$ (equivalently, $2^\kappa \simeq p$ for the $\mathbb{F}_p$ case) or (b) equal to $\kappa = N = t \cdot n$ (equivalently, $2^\kappa \simeq p^t$ for the $\mathbb{F}_p$ case). Since the cryptanalysis strategy of the three designs are very similar, in the following we give a complete analysis only for $\text{GMiMC}_{\mathsf{crf}}$, while we refer to [3, App. C - D] for the analysis of the other proposals.

4.1 Algebraic Attacks

In this section, we consider algebraic attacks against GMiMC. These attacks are particularly relevant for applications in which the attacker has access only to a limited number of (plaintext, ciphertext) pairs available to the attacker.

A main element in all these attacks is the degree reached in each of our constructions after r rounds. Here we give an idea of the analysis for the $\text{GMiMC}_{\mathsf{crf}}$ (similar for all other constructions).

For all (algebraic) attacks in the following, we only care about the minimum degree after round r on the $t-1$ branch:

$$d_{t-1,t-1} = 3^{r-2t+2}.$$

The degree of each word of the plaintext in the t-branch is given by a similar formula ([3, Sec. 4.1]) both for the univariate and multivariate cases.

GCD Attack. As for the original MiMC [4], an attack strategy against GMiMC is to compute the Greatest Common Divisors (GCD). In particular, given more than one known (plaintext, ciphertext) pair or even working on the output of different branches of a single known (plaintext, ciphertext) pair, one can construct their polynomial representations and compute their polynomial GCD to recover a multiple of the key.[9] Note that this is a known-plaintext attack, and not a chosen-plaintext one, and it is one of the few attacks that applies in the low-data scenario. *Since interpolation attack is more efficient than GCD attack (from the attacker point of view)*, we discuss all details of this attack in [3, Sec. 5.1 - App. C.4], together with other low-data attacks.

Gröbner Bases. The natural generalization of GCDs to the multivariate case is the notion of a Gröbner basis [25]. The attack proceeds like the GCD attack with the final GCD computation replaced by a Gröbner basis computation. Due to the Feistel structure, we highlight that it is possible to construct multivariate "meet-in-the-middle" polynomials. Here we only give the summary of the attack. The details of the attack is given in the extended version [3] of this article.

$\text{GMiMC}_{\mathsf{crf}}$ *(Case:* $2^\kappa \simeq p^t$*).* To prevent the Gröbner basis attack, the minimum number of rounds r must satisfy $p^\varphi \cdot \binom{t-\varphi+d-1}{d-1}^\omega \geq p^t$, for all $\varphi \in \{0, \ldots, t-2\}$, where the degree d is a function of the number of rounds r, that is, $d = d(r)$ and $2 \leq \omega < 3$ is the linear algebra constant. For our parameter choices, this expression is minimized for $\varphi = 0$. We thus require

$$\binom{t+d}{d}^\omega = \binom{t+3^{r-2t+2}}{3^{r-2t+2}}^\omega \approx p^t.$$

By setting $\omega = 2$ and after simplifying [3, Sec. 4.1] the above expression we obtain

$$r = \lceil 2t + 1/2 \, \log_2(p) \cdot \log_3 2 - 2 + \log_3 t \rceil .$$

To thwart Meet-in-the-Middle attacks, this value is doubled.

[9] Improving the computational complexity of this attack using more pairs is an open problem. Since the cost is dominated by the size of the polynomials involved, it is not clear if significant improvements are possible.

Interpolation Attack. As for the original MiMC, one of the most powerful attacks against the GMiMC family is the interpolation attack, introduced by Jakobsen and Knudsen [37] in 1997. This method can be extended to a key-recovery attack.

$\text{GMiMC}_{\mathsf{crf}}$ is secure against interpolation attack if $(3^{r-2t+2})^t \approx 2^N \simeq p^t$. Hence, $r = \frac{\log_2(p)}{\log_2 3} + (2t-2)$ rounds will be secure against the above-mentioned attacks. Conservatively, $2r+2$ rounds will be secure against meet-in-the-middle attacks/distinguishers for the case $2^\kappa \simeq p$, while $2r+t+1$ rounds will be secure against meet-in-the-middle attacks/distinguishers for the case $2^\kappa \simeq p^t$.

Higher-Order Differential. Let $\mathcal{A}$ be an affine space. Higher-order differential attacks [41] exploit the fact that $\bigoplus_{x \in \mathcal{A}} P(x) = 0$ if the dimension of $\mathcal{A}$ is higher than the degree of $P(\cdot)$. To thwart higher-order differential attacks, the number of rounds must be chosen in order to ensure that the algebraic degree of the GMiMC family of block ciphers is bigger than the biggest subspace in $\mathbb{F}$.

Due to the strategy exploited by the higher-order differential attack, there is a crucial difference between the cases $\mathbb{F}_{2^N}$ and $\mathbb{F}_p$. In particular, while $\mathbb{F}_{2^m}$ is always a subspace of $\mathbb{F}_{2^n}$ for each $m \leq n$, the only subspaces of $\mathbb{F}_p$ are $\{0\}$ and $\mathbb{F}_p$. It follows that the biggest subspace of $\left(\mathbb{F}_p\right)^t$ has dimension t, with respect to the biggest subspace of $\left(\mathbb{F}_{2^n}\right)^t$, which has dimension $n \cdot t = N$. As shown in details in [3], the number of rounds previously given (necessary to protect GMiMC w.r.t. previous attacks) guarantees security against this attack.

4.2 Statistical Attacks

All the statistical attacks which we have considered can be carried out in the same way over $\mathbb{F}_p$ or/and over $\mathbb{F}_{2^n}$ against reduced round GMiMC. We analyzed the security of GMiMC against *classical and truncated differential, linear*, and *impossible differential* attacks. Since this type of attacks does not provide more advantage compared to the algebraic attacks, we skip the details. We refer the interested readers to the extended version of this article [3] for the details of the aforementioned analyses.

4.3 Other Attacks

We claim that GMiMC instantiated using the number of rounds of the univariate case[10] is secure in the known- and chosen-key model. In particular, such permutation is used in order to construct the hash function using the sponge construction. We recall that the (required) indifferentiable of the internal permutation of a sponge function from a random oracle - for a fixed key - is equivalent to the security of GMiMC in the known- and chosen-key model.

[10] The number of rounds in this case is given considering the number of rounds of any possible distinguisher - which is independent of the secret key - in the MitM scenario plus a secure margin. Since the key is fixed in the known- and chosen-key model, this number of rounds provides the security in these scenarios.

Finally we explicitly state that we do not have claims in the related-key model as we do not consider it to be relevant for the intended use case.

Quantum Improvements. In a post-quantum setting, the cost of exhaustive key search is square rooted by Grovers' algorithm. Statistical attacks remain unchanged (except perhaps their computational part). The quantum interpolation attack gives no significant advantage to the adversary since the attack requires $d/2$ queries, where d is the degree of the polynomial [24]. It is not clear that Grover's algorithm can help to improve the GCD attack. The attack cost $\mathcal{O}(d \cdot \log^2 d)$ operations on inputs of size d. Thus, even with the square root reduction the attacker will still need to write the inputs of size d as classically; a similar argument holds for Gröbner basis attacks.

Finally, since we are here interested in post-quantum security of classical schemes and not in the security of symmetric primitives running on a quantum computer themselves, better attacks are known using Simon's algorithm [33].

5 Parameter-Space Exploration

We compared the effects of different parameters in our Feistel-based constructions with block size N. We compared the parameters of the GMiMC whithin the range of the values in the three different applications e.g. length of the prime or n, number of branches t and number of rounds R. The main purpose of this comparison is to identify the optimal range of values for these parameters when the block size is fixed.

Both $\text{GMiMC}_{\mathsf{crf}}$ and $\text{GMiMC}_{\mathsf{erf}}$ have only one multiplication at each round while $\text{GMiMC}_{\mathsf{mrf}}$ has $t/2$ multiplications per round. By our analysis, it turns out that $\text{GMiMC}_{\mathsf{erf}}$ is always more efficient than $\text{GMiMC}_{\mathsf{crf}}$ and $\text{GMiMC}_{\mathsf{mrf}}$, since it always requires a lower number of rounds to be secure. In this article we only provide the performance of the best candidate for the target applications. For a more generic discussion of the different parameters and their effect on the multiplicative complexity we refer to the [3, Sec. 6].

6 Implementation results

6.1 MPC Setting

Security Model. Our protocols are built to support the SPDZ-family protocols which guarantee security even when there is a dishonest majority of parties involved in the computation [13, 28]. This means that we support an arbitrary number n of computing parties and an adversary can corrupt up to $n-1$ parties.

The implementation is written in a high-level language similar to Python [40] and is compatible with MP-SPDZ [7] and SCALE-MAMBA [6]. We have benchmarked the protocols using the SPDZ framework [11], which provides active security against multiple malicious parties.

[11] `https://github.com/bristolcrypto/SPDZ-2`

To compute a circuit with secret shared inputs in SPDZ, there are two generic phases. The first step is to produce random Beaver triples, also called the preprocessing phase, which is independent of the inputs and can be done in advance. The second step is the online phase, which consumes a triple whenever there is a multiplication between shared values. Additions of secret values and scalar multiplications are (almost) for free in SPDZ. The protocols ran across two computers with Intel i7-4790 CPUs at 3.60GHz and 16GB of RAM connected via a 1 GB/s LAN network and an average round-trip time of 0.3 ms (see Table 1). In our setting, both keys and messages are secret shared between the two parties and each experiment was averaged among five executions with at least 1000 block cipher calls.

Mode	(t,R)	Online cost				Prep (ms)
		# Comm. rounds	Openings	Latency (ms)/$\mathbb{F}_p$	Throughput $\mathbb{F}_p$/s	
$\text{GMiMC}_{\text{crf}}$	4	178	534	3.65	15026	2.96
$\text{GMiMC}_{\text{erf}}$		172	516	3.55	15669	2.86
$\text{GMiMC}_{\text{mrf}}$		175	525	3.62	8194	5.83
MiMC	4 blocks	73	876	1.58	9965	4.86
$\text{GMiMC}_{\text{crf}}$	16	238	714	1.21	39247	0.99
$\text{GMiMC}_{\text{erf}}$		208	624	1.06	49006	0.86
$\text{GMiMC}_{\text{mrf}}$		183	549	1.02	8440	6.1
MiMC	16 blocks	73	3504	0.47	10780	4.86

Table 1: Two-party costs for MiMC and GMiMC over a 1Gb/s LAN network with an average ping time of 0.3ms. The variable t denotes the number of branches for GMiMC and no. of blocks for MiMC, whereas R is the number of cipher rounds.

For a complete measurement of an MPC protocol, one needs to have in mind both preprocessing and online phases. The preprocessing phase cost is determined by the number of secret shared multiplications. Performance of the online phase is given by the multiplicative depth of the circuit to be evaluated as well as the number of openings (whenever a party reveals a secret value). For the online phase we give measurements in terms of *latency* and *throughput*. Latency indicates the minimum time spent for computing one encrypted $\mathbb{F}_p$ block, whereas throughput shows the maximum $\mathbb{F}_p$ objects that can be encrypted in parallel per second. Since the only non-linear operation we use in our block ciphers is $x \mapsto x^3$, this is done with three openings and two Beaver triples (for details see [49]). We instantiate each block cipher with 8 and 64 input blocks/branches, where each block lies in $\mathbb{F}_p$ and $p \approx 2^{128}$. Note that for GMiMC constructions in MPC we have used an n-bit key. For a fair comparison with previous evaluations of MiMC in SPDZ, the online phase runs on a single thread.

The preprocessing column (Prep, Table 1) denotes the amount of time required to generate the triples for a single block cipher evaluation (4 or 16 en-

crypted blocks/branches) in a two party SPDZ protocol. The figures for this column were estimated using the recent protocol by Keller et al. [39] which is the fastest known protocol for SPDZ triples. We used the LowGear protocol with computational security 128 and 64 bit statistical security.

Experiments (Table 1) show that $\text{GMiMC}_{\mathsf{crf}}$ and $\text{GMiMC}_{\mathsf{erf}}$ have a very fast preprocessing phase because they perform a low number of multiplications. A big advantage of these two is how well they scale in terms of triples used, since they require one multiplication per cipher iteration. This is in contrast with MiMC, where increasing the number of blocks to be encrypted by a factor of c results in c times more multiplications. We stress that these two constructions are first to our knowledge which avoid the linear increase of pre-processing data with the number of blocks. To give an example of this behavior consider the case of 16 blocks for the preprocessing column (Table 1) for $\text{GMiMC}_{\mathsf{erf}}$ is 5.5 times smaller than MiMC: 0.86ms vs. 4.86ms. We can see that $\text{GMiMC}_{\mathsf{crf}}$ and $\text{GMiMC}_{\mathsf{erf}}$ have a higher online throughput compared to the rest of the variants, although they have a higher larger number of communication rounds. The reason is that fewer openings - or multiplications in our case - mean less data sent between the parties so we can batch more executions in parallel. Thus in a LAN network the number of rounds has a minor impact. As for the WAN results, as expected MiMC preprocessing phase induces a large cost but the online phase is slightly faster than our proposed ciphers. The interested reader can find the experiments on a slow network in the full version of our paper [3].

6.2 SNARKs

The rank-1 constraints (r1cs) in SNARK is defined in [12] as a system of bilinear equations over a field $\mathbb{F}$. The number of *rank-1 constraints* (r1cs) for a function contributes to the efficiency of the SNARK algorithm [4]. In this setting we count the number of multiplications required to generate the values of *witness variables* defined in [12]. We describe the r1cs for $\text{GMiMC}_{\mathsf{crf}}$ in Appendix A. For the other two construction it can be constructed similarly.

We implemented all three constructions in a SNARK setting using NTL [52] for the permutations and hash functions. $\text{GMiMC}_{\mathsf{erf}}$ shows the best performance among the 3 constructions and we only show its performance result in Table 2. We also compared the performance with MiMC. For $N \approx 1024$-bit (prime) block size $\text{GMiMC}_{\mathsf{erf}}[N, t, n]$, where $t = 8$, shows some improvement over MiMC-1025. For hashing a single message block, GMiMCHash-256 is more than 1.2 times faster than MiMCHash-256 and is significantly (> 12 times) faster than SHA-256. We stress that in comparison with MiMCHash the primary advantage of $\text{GMiMC}_{\mathsf{erf}}$Hash is that it can be used over 256 bit or smaller field size. For all the field operations we have used the NTL together with the `gf2x` library. All the computations were performed on a system having an Intel Core i7-4790 with 3.6 GHz processor with 16 GB memory. We took the average time over ≈ 2000 iterations. The last column in the Table 2 is mainly to demonstrate the performance of the design in a smaller field (below 128 bit prime).

	MiMC [4]		$\text{GMiMC}_{\mathsf{erf}}$		
$(t, \log_2(p), R)$	$(1, 1024, 646)$	$(2, 513, 647)$	$(4, 256, 332)$	$(8, 128, 178)$	$(16, 64, 141)$
constraint generation	4.553 ms	5.077 ms	4.735 ms	4.732 ms	8.057 ms
witness generation	1.079 ms	0.639 ms	0.388 ms	0.296 ms	0.449 ms
total time	5.632 ms	5.716 ms	5.123 ms	5.028 ms	8.507 ms
#additions	646	1293	996	1246	2115
#multiplications	1293	1293	664	356	282

Table 2: Comparison of MiMC with $\text{GMiMC}_{\mathsf{erf}}$ (with different numbers of branches) in SNARK in $\mathbb{F}_p$ when the block size is 1024 bits.

Note that the number of constraints for GMiMCHash-256 is only one more than the number of constraints for $\text{GMiMC}_{\mathsf{erf}}$. Hence the time taken by the hash function and the permutation with fixed key are the same (in Table 2).

6.3 Post-Quantum Signatures

Picnic [22] is a new class of digital signature schemes which derive their security entirely from the security of symmetric-key primitives, have extremely small key pairs, and are highly parameterizable. The construction is based on a one-way function f, where for the secret key x the image $y = f(x)$ is published as the public key. A signature on a message is then obtained from a non-interactive zero-knowledge proof of the relation $y = f(x)$, that incorporates the message in the challenge generation. This proof uses ZKB++, a Σ-protocol for statements over general circuits made non-interactive. When instantiating f with LowMC [2, 5], reducing the signature size by reducing the number of multiplication gates comes at the cost of a more expensive linear layer, which leads to a runtime vs. signature size trade-off. Since the security proofs in [22] only require a block cipher with a reduced data complexity of 1, the overall performance can be greatly improved as this fact allows to choose instances with less rounds. For the 128-bit PQ security level (i.e., 256-bit block size and key size) a good trade-off can be found by using 10 S-Boxes and 38 rounds, resulting in a view size of 1140 bits.

We implemented the signature scheme using $\text{GMiMC}_{\mathsf{erf}}$ with key size and block size of ≈ 256 bits to build the one-way function. We consider instances with a data complexity of 1. The reduction steps of the modular multiplications were accelerated by using special prime moduli and irreducible polynomials of special form for prime fields and binary fields, respectively: generalized Mersenne primes [53] were used for prime fields and trinomials and pentanomials with middle terms close to each other [50] were used for binary fields.

In Table 3, we compare the circuit runtimes (i.e., runtimes without protocol overheads such as pseudo-random number sampling and the computation of commitments) of MiMC and $\text{GMiMC}_{\mathsf{erf}}$ with different numbers of branches benchmarked on an Intel Core i7-4790 with 3.6 GHz. We also include the view

Scheme	(n, t, R)	Sign	Verify	View size
MiMC [4]	$(256, 1, 162)$	333.97 ms	166.28 ms	83456 bits
	$(272, 1, 172)$	92.45 ms	46.32 ms	94112 bits
$\text{GMiMC}_{\mathsf{erf}}$ over $\mathbb{F}_p$	$(3, 86, 261)$	97.32 ms	72.06 ms	1566 bits
	$(4, 64, 196)$	62.35 ms	45.16 ms	1568 bits
	$(32, 8, 55)$	4.95 ms	3.05 ms	3520 bits
	$(136, 2, 163)$	67.51 ms	35.21 ms	44336 bits
$\text{GMiMC}_{\mathsf{erf}}$ over $\mathbb{F}_{2^n}$	$(3, 86, 261)$	16.06 ms	10.76 ms	**783 bits**
	$(17, 16, 63)$	3.73 ms	2.30 ms	1071 bits
	$(33, 8, 56)$	**3.34 ms**	**2.29 ms**	1848 bits
LowMC-$(256, 10, 38)$	-	3.74 ms	3.52 ms	1140 bits
LowMC-$(256, 1, 363)$	-	9.55 ms	7.12 ms	1089 bits

Table 3: Comparison of MiMC with $\text{GMiMC}_{\mathsf{erf}}$ and LowMC [5] when the block size is ≈ 256 bits in the context of ZKB++. In LowMC-(N, m, R), N denotes the block size, m is the number of S-Boxes, and R denotes the number of rounds. Runtimes given for Sign and Verify are for the circuit computations only.

size required per repetition of ZKB++, and numbers for two instances of LowMC using optimizations for the round key computations and linear layer [31]. Measuring only the circuit runtimes allows us to obtain a more accurate comparison in terms of computation time, which directly relates to the total runtime of the protocol, whereas the view sizes directly related and to the signature size.

Instantiations using $\mathbb{F}_{2^n}$ tend to perform better than the comparable parameterizations in $\mathbb{F}_p$ for mainly two reasons: in $\mathbb{F}_{2^n}$ additions do not require reductions, the cubing operation can be implemented with only one multiplication. In any case, even for very small fields with slower runtime, $\text{GMiMC}_{\mathsf{erf}}$ performs significantly better in terms of view size and runtime than MiMC. Compared to LowMC, choosing an instance over $\mathbb{F}_{2^3}$ allows us to beat the smallest signatures sizes obtainable using LowMC with one S-Box by 306 bits in terms of view size. We also note that both signing and verification times are smaller when using instances providing a good trade-off (i.e., setting $n = 17$ or $n = 33$), and view sizes can be kept small too.

7 Discussion and Open Problems

One key take-away of this work is that, when it comes to building structures in symmetric cryptography with MPC and related applications in mind, there are old and prematurely discarded ideas that are worthwhile to revisit.

Unbalanced Feistel networks appeared in the late 1980s and have rarely been used in recent designs. As an illustration, consider that among all the lightweight

block cipher designs listed on the CryptoLux lightweight block cipher wiki,[12] 7 are Type-II GFNs and 10 are balanced Feistel networks, whereas *none* is of the UFN or ERF type. And yet exactly those types turn out to be the best in our setting. The structure of MiMC is strongly related to a design from the mid 1990s, which in recent textbooks [42, Sect. 8.4] was even shown as an example of how not to design a cipher. However, it has turned out to be very good in many applications where multiplicative complexity matters. It may well be that the Cryptographers had lost interest in the UBF or never considered it a reasonable option and yet it is the best in several of our specific use cases. This naturally raises the question: What are other known but out-of-fashion structures which might be very suitable for MPC, SNARKs, PQ signatures or related applications?

In this paper our focus was on constructions that can natively deal with elements of fields in large characteristic. For other use-cases, binary extension fields may be interested. For this, we leave a general security analysis (especially concerning higher-order differential attacks) as an open future problem.[13]

References

1. S. Agrawal, P. Mohassel, P. Mukherjee, and P. Rindal. DiSE: Distributed symmetric-key encryption. In Lie et al. [43], pages 1993–2010.
2. M. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. Cryptology ePrint Archive, Report 2016/687, 2016. `http://eprint.iacr.org/2016/687`.
3. M. R. Albrecht, L. Grassi, L. Perrin, S. Ramacher, C. Rechberger, D. Rotaru, A. Roy, and M. Schofnegger. Feistel structures for mpc, and more. Cryptology ePrint Archive, Report 2019/397, 2019. `https://eprint.iacr.org/2019/397`.
4. M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 191–219. Springer, Heidelberg, Dec. 2016.
5. M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, Apr. 2015.
6. A. Aly, M. Keller, E. Orsini, D. Rotaru, P. Scholl, N. Smart, and T. Wood. Scale-mamba v1.3 : Documentation, 2018. `https://homes.esat.kuleuven.be/~nsmart/SCALE/`.
7. N. Analytics. MP-SPDZ, 2019. `https://github.com/n1analytics/MP-SPDZ`.
8. B. Applebaum, N. Haramaty, Y. Ishai, E. Kushilevitz, and V. Vaikuntanathan. Low-Complexity Cryptographic Hash Functions. In *8th Innovations in Theoretical Computer Science Conference – ITCS 2017*, volume 67 of *LIPIcs*, pages 7:1–7:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
9. B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[12] `https://www.cryptolux.org/index.php/Lightweight_Block_Ciphers`

[13] Our ZKBoo use-case uses such fields, but the analysis we provide is rather specific to the needed low data-complexity security requirements

10. D. W. Archer, D. Bogdanov, L. Kamm, Y. Lindell, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright. From keys to databases – real-world applications of secure multi-party computation. Cryptology ePrint Archive, Report 2018/450, 2018. `https://eprint.iacr.org/2018/450`.
11. E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.
12. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, Aug. 2013.
13. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 169–188. Springer, Heidelberg, May 2011.
14. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the indifferentiability of the sponge construction. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, Apr. 2008.
15. E. Biham, R. Anderson, and L. Knudsen. Serpent: A new block cipher proposal. In *FSE 1998*, pages 222–238, 1998.
16. D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In S. Jajodia and J. López, editors, *ESORICS 2008*, volume 5283 of *LNCS*, pages 192–206. Springer, Heidelberg, Oct. 2008.
17. D. Boneh, S. Eskandarian, and B. Fisch. Post-quantum EPID signatures from symmetric primitives. In *CT-RSA*, volume 11405 of *Lecture Notes in Computer Science*, pages 251–271. Springer, 2019.
18. D. Boneh, Y. Ishai, A. Passelègue, A. Sahai, and D. J. Wu. Exploring crypto dark matter. In *Theory of Cryptography Conference*, pages 699–729. Springer, 2018.
19. J. Boyar, R. Peralta, and D. Pochuev. On the multiplicative complexity of Boolean functions over the basis (cap, +, 1). *Theor. Comput. Sci.*, 2000.
20. A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In T. Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 313–333. Springer, Heidelberg, Mar. 2016.
21. A. Chailloux. Quantum security of the fiat-shamir transform of commit and open protocols. *IACR Cryptology ePrint Archive*, 2019:699, 2019.
22. M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 17*, pages 1825–1842. ACM Press, Oct. / Nov. 2017.
23. M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. The Picnic Signature Algorithm Specification, 2017. `https://github.com/Microsoft/Picnic/blob/master/spec.pdf`.
24. A. M. Childs, W. van Dam, S. Hung, and I. E. Shparlinski. Optimal quantum algorithm for polynomial interpolation. In *ICALP*, volume 55 of *LIPIcs*, pages 16:1–16:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
25. D. A. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra (2. ed.)*. Undergraduate texts in mathematics. Springer, 1997.
26. J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen. Nessie Proposal: NOEKEON, 2000. `http://gro.noekeon.org/Noekeon-spec.pdf`.

27. I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen. Asynchronous multiparty computation: Theory and implementation. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 160–179. Springer, Heidelberg, Mar. 2009.
28. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, Aug. 2012.
29. D. Derler, S. Ramacher, and D. Slamanig. Generic double-authentication preventing signatures and a post-quantum instantiation. In J. Baek, W. Susilo, and J. Kim, editors, *ProvSec 2018*, volume 11192 of *LNCS*, pages 258–276. Springer, Heidelberg, Oct. 2018.
30. D. Derler, S. Ramacher, and D. Slamanig. Post-Quantum Zero-Knowledge Proofs for Accumulators with Applications to Ring Signatures from Symmetric-Key Primitives. In *Post-Quantum Cryptography – PQCrypto 2018*, volume 10786 of *LNCS*, pages 419–440. Springer, 2018.
31. I. Dinur, D. Kales, A. Promitzer, S. Ramacher, and C. Rechberger. Linear equivalence of block ciphers with partial non-linear layers: Application to lowmc. In *EUROCRYPT (1)*, volume 11476 of *Lecture Notes in Computer Science*, pages 343–372. Springer, 2019.
32. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. *IACR Cryptology ePrint Archive*, 2019:190, 2019.
33. X. Dong, Z. Li, and X. Wang. Quantum cryptanalysis on some generalized Feistel schemes. Cryptology ePrint Archive, Report 2017/1249, 2017. `https://eprint.iacr.org/2017/1249`.
34. Y. Doröz, A. Shahverdi, T. Eisenbarth, and B. Sunar. Toward practical homomorphic evaluation of block ciphers using prince. In R. Böhme, M. Brenner, T. Moore, and M. Smith, editors, *FC 2014 Workshops*, volume 8438 of *LNCS*, pages 208–220. Springer, Heidelberg, Mar. 2014.
35. L. Grassi, C. Rechberger, D. Rotaru, P. Scholl, and N. P. Smart. MPC-friendly symmetric key primitives. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 430–443. ACM Press, Oct. 2016.
36. V. Grosso, G. Leurent, F.-X. Standaert, and K. Varici. LS-designs: Bitslice encryption for efficient masked software implementations. In C. Cid and C. Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 18–37. Springer, Heidelberg, Mar. 2015.
37. T. Jakobsen and L. R. Knudsen. The interpolation attack on block ciphers. In E. Biham, editor, *FSE'97*, volume 1267 of *LNCS*, pages 28–40. Springer, Heidelberg, Jan. 1997.
38. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In Lie et al. [43], pages 525–537.
39. M. Keller, V. Pastro, and D. Rotaru. Overdrive: Making SPDZ Great Again. In *Advances in Cryptology – EUROCRYPT 2018*, volume 10822 of *LNCS*, pages 158–189. Springer, 2018.
40. M. Keller, P. Scholl, and N. P. Smart. An architecture for practical actively secure MPC with dishonest majority. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 549–560. ACM Press, Nov. 2013.
41. L. R. Knudsen. Truncated and higher order differentials. In B. Preneel, editor, *FSE'94*, volume 1008 of *LNCS*, pages 196–211. Springer, Heidelberg, Dec. 1995.

42. L. R. Knudsen and M. J. B. Robshaw. *The Block Cipher Companion*. Springer Publishing Company, Incorporated, 2011.
43. D. Lie, M. Mannan, M. Backes, and X. Wang, editors. *ACM CCS 18*. ACM Press, Oct. 2018.
44. P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 311–343. Springer, Heidelberg, May 2016.
45. M. Naehrig, K. E. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011*, pages 113–124, 2011.
46. K. Nyberg. Generalized Feistel networks. In K. Kim and T. Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 91–104. Springer, Heidelberg, Nov. 1996.
47. I. S. M. S. Overview. `https://gist.github.com/maxvt/bb49a6c7243163b8120625fc8ae3f3cd`.
48. Partisia. `https://partisia.com/`.
49. D. Rotaru, N. P. Smart, and M. Stam. Modes of operation suitable for computing on encrypted data. *IACR Trans. Symm. Cryptol.*, 2017(3):294–324, 2017.
50. M. Scott. Optimal irreducible polynomials for GF(2^m) arithmetic. Cryptology ePrint Archive, Report 2007/192, 2007. `http://eprint.iacr.org/2007/192`.
51. Sepior. `https://sepior.com/`.
52. V. Shoup. Number Theory Library 5.5.2 (NTL). `http://www.shoup.net/ntl/`.
53. J. A. Solinas. Generalized mersenne numbers. Technical report, NSA, 1999.
54. T. Suzaki and K. Minematsu. Improving the generalized Feistel. In S. Hong and T. Iwata, editors, *FSE 2010*, volume 6147 of *LNCS*, pages 19–39. Springer, Heidelberg, Feb. 2010.
55. T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. TWINE : A lightweight block cipher for multiple platforms. In L. R. Knudsen and H. Wu, editors, *SAC 2012*, volume 7707 of *LNCS*, pages 339–354. Springer, Heidelberg, Aug. 2013.
56. Unbound. `https://www.unboundtech.com/`.

A R1CS for GMiMC$_{\mathsf{crf}}$

For GMiMC$_{\mathsf{crf}}$ the rank-1 constraints are as follows:

$$\sum_{i=0}^{t-1} X_i + U + k_r + C_r = 0,\ U \cdot U = Y,\ U \cdot Y + X_{t-1} = Z,$$

where k_r and C_r are round keys and round constants respectively. For GMiMC$_{\mathsf{crf}}$ Hash the round keys are fixed to a constant. The number of multiplication for GMiMC$_{\mathsf{crf}}$ Hash is 2 per round. Therefore the total number of multiplications is $2R$ where R is the number of rounds in the block cipher GMiMC$_{\mathsf{crf}}$. Each round also requires $t-1$ field additions.