MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity

Martin Albrecht[1], Lorenzo Grassi[3], Christian Rechberger[2,3], Arnab Roy[2], and Tyge Tiessen[2]

[1] Royal Holloway, University of London, UK
martinralbrecht@googlemail.com
[2] DTU Compute, Technical University of Denmark, Denmark
{arroy,crec,tyti}@dtu.dk
[3] IAIK, Graz University of Technology, Austria
{christian.rechberger,lorenzo.grassi}@iaik.tugraz.at

Abstract. We explore cryptographic primitives with low multiplicative complexity. This is motivated by recent progress in practical applications of secure multi-party computation (MPC), fully homomorphic encryption (FHE), and zero-knowledge proofs (ZK) where primitives from symmetric cryptography are needed and where linear computations are, compared to non-linear operations, essentially "free". Starting with the cipher design strategy "LowMC" from Eurocrypt 2015, a number of bit-oriented proposals have been put forward, focusing on applications where the multiplicative depth of the circuit describing the cipher is the most important optimization goal.

Surprisingly, albeit many MPC/FHE/ZK-protocols natively support operations in GF(p) for large p, very few primitives, even considering all of symmetric cryptography, natively work in such fields. To that end, our proposal for both block ciphers and cryptographic hash functions is to reconsider and simplify the round function of the Knudsen-Nyberg cipher from 1995. The mapping $F(x) := x^3$ is used as the main component there and is also the main component of our family of proposals called "MiMC". We study various attack vectors for this construction and give a new attack vector that outperforms others in relevant settings.

Due to its very low number of multiplications, the design lends itself well to a large class of applications, especially when the depth does not matter but the total number of multiplications in the circuit dominates all aspects of the implementation. With a number of rounds which we deem secure based on our security analysis, we report on significant performance improvements in a representative use-case involving SNARKs.

Keywords: distributed cryptography, cryptanalysis, block ciphers, hash functions, zero knowledge

1 Introduction

Modern cryptography developed many techniques that go well beyond solving traditional confidentiality and authenticity problems in two-party communication. Secure multi-party computation (MPC), zero-knowledge proofs (ZK), and fully homomorphic encryption (FHE) are some of the most striking examples. In various applications of these three technologies, part of the circuit or function that is being evaluated is in turn a cryptographic primitive such as a PRF, a symmetric encryption scheme, or a collision resistant function.

In this work, we focus on a large class of such applications where the total number of field multiplications in the underlying cryptographic primitive poses the largest performance bottleneck. Examples include MPC protocols based on Yao's garbled circuit and all ZK-proof system that we are aware of, including recent developments around SNARKs [BSCG+13] which found practical applications, e.g., in Zerocash [BCG+14]. This motivates the following question addressed in this work: *How could a construction for a secure block cipher or a secure cryptographic hash functions look like that minimizes the number of field multiplications?*

Earlier work on specialized designs for such applications, like LowMC [ARS+15], Kreyvium [CCF+16], or the very recent FLIP [MJSC16] all consider the case of *Boolean* multiplications and mostly focus on the depth of the resulting circuit.

Surprisingly, albeit many MPC/FHE/ZK-protocols natively support operations in GF(p) for large p, very few candidates, even considering all of symmetric cryptography, exist which natively work in such fields. Our focus in this paper is hence on *multiplications in the larger fields* GF(2^m) and GF(p) which is motivated as follows: As many protocols support multiplications in larger fields natively, encoding of a description in GF(2) is cumbersome and inefficient. Whilst it is possible to do bit operations over $\mathbb{F}_p$ using standard tricks (which turn XOR into a non-linear operation), such a conversion is expensive. Consider AES as an example: it allows for an efficient description in a variety of field sizes. This is also the reason why the bit-based LowMC, which has a lower number of AND gates, can often barely, if at all, outperform AES in actual implementations of the GMW MPC protocols, despite being much better than AES in terms of GF(2) metrics. See [ARS+16, Table 6] for details of the most striking example. This is also partly due to the *very* high number of XORs computed in LowMC resulting them to be no longer negligible.

Contributions and related work. The design we propose is extremely simple: A function $F(x) := x^3$ is iterated with subkey additions. This is described in detail in Section 2. In fact, our design is a simplified variant of a design by Nyberg and Knudsen [KN95] from the 1990s, which was aimed to demonstrate ways to achieve provable security against the then emerging differential and linear attacks, using a small number of rounds (smaller than, say, DES). However, not much later, [JK97] showed very efficient, even practical interpolation attacks

on such proposals. Indeed, our proposal resembles $\mathcal{PURE}$, a design introduced in [JK97] in order to present their attack. We pick up this work from almost 20 years ago and study in earnest if a much higher number of rounds can make this design secure in Section 4. It turns out, perhaps surprisingly, that the required much higher number of rounds (in the order of 100s instead of 10 or less) is *very competitive* when it comes to the new application areas of symmetric cryptography that motivate this work.

We propose several variants of our design called MiMC: variants for GF(p) and GF(2^n) as well as variants that use the cube mapping directly or in a Feistel structure. MiMC can be used for encryption as well as for collision-resistant cryptographic hashing. See Section 2 for the basic variant in GF(2^n) and Section 5 for a discussion on the other variants. MiMC is distinguished from any of the many constructions that have been proposed in this field recently to the extent that it contradicts popular belief: A recent standard textbook [KR11, Sect. 8.4] explicitly considers such constructions as "not serious, for various reasons".

Metrics. Given the wide variety of applications and protocols, no simple metric will be able to reliably predict application level performance. Issues of conversion between various field types (as the conversion between GF(2) and GF(p) mentioned above, which can be quite costly) add to the complication. Nevertheless, in order to give at least some hint towards expected performance, we will use the minimal number of multiplication to compute an output (minMULs), and the average number of multiplications needed per input bit (MULs/bit) on various designs. For the important special case of GF(2) we will use minANDs and ANDs/bit, respectively.

A discussion of various constructions in GF(p) and GF(2) can be found in Section 3. In the benchmarking part in Section 6.1, we will also come across the case of an extremely imbalanced LowMC-variant where this simple metric clearly fails to predict actual performance. In Appendix B we will also see that the application performance is not independent of the size of the multiplier, but for the sizes relevant for MiMC this dependence is fairly weak.

Implementation Results. The hashing mode for GF(p) may prove to be particularly useful as it is the first of its kind, despite various applications in verifiable computing [CFH+15] and applications of SNARKS like Zerocash [BCG+14] requiring such a function. Due to a lack of an alternative, authors implemented and optimized SHA-256, which leads to a bottleneck in efficiency. We demonstrate that MiMC compares very favorably in such an application. Based on our experiments and implementations, we report a factor 10 improvement in Section 6.1. We briefly mention more direct implementations in Section 6.2 and discuss the suitability of the design for cheap (generic) protection against higher-order side-channel attacks in Section 6.3.

In follow-up to this work [GRR+16], it was found that MiMC is also a very competitive candidate as an MPC-friendly PRF. Compared to AES, benchmark results showed that MiMC has a more than 10 times higher throughput in the

online phase, and still about six times faster in the offline/precomputation phase in the LAN setting. Even the latency, which one could expect to be relatively high for MiMC due to its serial nature and the relatively high number of rounds, is better than the latency of AES. Note that for the AES case, this does not include conversion losses due to the application not using the AES field GF(2^8), and hence the difference in real-world application settings will likely be larger.

2 The MiMC primitives

In the following, we describe a block cipher, a permutation, and a permutation-based cryptographic hash function with a low number of multiplications in a finite field $\mathbb{F}_q$ (alternatively GF(q)) where q is either a prime p or a power of 2.

2.1 The block cipher

In order to achieve an efficient implementation over a field $\mathbb{F}_q$ (with q either prime or a power of 2), i.e., to minimize computationally expensive multiplications in the field, our design operates entirely over $\mathbb{F}_q$, thereby avoiding S-boxes completely. More precisely, we use a permutation polynomial over $\mathbb{F}_q$ as round function. In the following, we restrict ourselves to $\mathbb{F}_{2^n}$ and we denote by MiMC-b/κ a keyed permutation with block size b and key size κ. The concept however equally applies to $\mathbb{F}_p$, which we will discuss briefly in Section 5.

MiMC-n/n. Our block cipher is constructed by iterating a round function r times where each round consists of a key addition with the key k, the addition of a round constant $c_i \in \mathbb{F}_{2^n}$, and the application of a non-linear function defined as $F(x) := x^3$ for $x \in \mathbb{F}_{2^n}$. For a discussion of this particular choice of polynomial and alternatives, we refer to Section 5.3. The ciphertext is finally produced by adding the key k again to the output of the last round. Hence, the round function is described as $F_i(x) = F(x \oplus k \oplus c_i)$ where $c_0 = c_r = 0$ and the encryption process is defined as

$$E_k(x) = (F_{r-1} \circ F_{r-2} \circ \dots F_0)(x) \oplus k.$$

We choose n to be odd and the number of rounds as $r = \left\lceil \frac{n}{\log_2 3} \right\rceil$. The $r-1$ round constants are chosen as random elements from $\mathbb{F}_{2^n}$.

Note that the random constants c_i do not need to be generated for every evaluation of MiMC. Instead the constants are fixed once and can be hard-coded into the implementation on either side. No extra communication is thus needed, just as with round constants in LowMC, AES, or in fact any other cipher.

Decryption for MiMC-n/n can be realized analogously to encryption by reversing the order of the round constants and using $F^{-1}(x) := x^s$ with $s = (2^{n+1} - 1)/3$ instead of $F(x) := x^3$ (the complete derivation of s is given in Sect. 4, Lemma 1). Hence, encryption and decryption need to be implemented

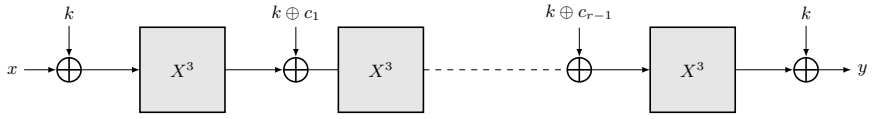

Fig. 1: r rounds of MiMC-n/n

separately. Furthermore, decryption is much more expensive than encryption. Using modes where the inverse is not needed is thus advisable. We note that for our targeted applications, such as PRFs or cryptographic hash functions, computing the inverse is usually not required. We therefore provide benchmark results only for the encryption function. The fact that the inverse has a more complex algebraic description also has a beneficial effect on security as it limits cryptanalytic approaches that try to combine the encryption and decryption direction, such as inside-out approaches.

MiMC-2n/n (Feistel). By using the same non-linear permutation in a Feistel network, we can process larger blocks at the cost of increasing the number of rounds by a factor of two. The round function of MiMC-$2n/n$ is defined as following

$$x_L \| x_R \longleftarrow x_R \oplus (x_L \oplus k \oplus c_i)^3 \| x_L.$$

The round constants c_i are again random elements of $\mathbb{F}_{2^n}$ except for the first and last round constants which are equal to 0. In the last round, the swap operation is not applied. The number of rounds for the Feistel version is $r' = 2 \cdot r = 2 \cdot \left\lceil \frac{n}{\log_2 3} \right\rceil$, where r is the number of rounds of MiMC-n/n.

Decryption for MiMC-$2n/n$ can easily be realized by using the encryption function with reversed order of round constants, as usual for Feistel networks.

2.2 The permutation

To construct the permutation MiMC^P from the cipher MiMC as described above, we simply set the key to the all-0 string.

2.3 The hash function

For the hash function MiMChash, we propose to use the permutation MiMC^P in the sponge framework [BDPA08]. Given a permutation of size n, and a desired security level s, we can hash $r = n - 2s$ bits per call to the permutation. The MiMC permutation can be realized in both variants (Feistel and non-Feistel) by setting the key to 0^κ where κ is the size of the key in bits. MiMCHash-ℓ denotes the hash function with ℓ bit output.

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 MiMCHash-t we use MiMC-n/n permutation where $n = 4 \cdot t+1$ and $s = 2 \cdot t$. For MiMCHash-256 we thus use a MiMC-n/n permutation with $n = 1025$. 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 of 128-bits and preimage security of 256-bits, and in contrast to SHA-256 also full 256-bit 2nd-preimage security independent of the message length. We also propose MiMCHash-256b, which also offers collision resistance of 128 bits but only 128-bit security against preimage-style attacks, similar to SHAKE-256 as specified in the new SHA-3 standard. This construction makes use of a MiMC-n/n permutation where $n = 769$. The rate and the capacity are chosen as 512 and 257 respectively. More generally for MiMCHash-tb, we use the MiMC-n/n permutation where $n = 3 \cdot t + 1$ and $s = t + 1$.

3 Related designs and comparison

In this section, we give an overview of related designs, i.e. symmetric primitives which are based on arithmetic operations in some ring.

3.1 Knudsen-Nyberg cipher

As discussed above, our design can be seen as a resurrection of a design due to Knudsen and Nyberg in [KN95], who proposed a DES-like cipher using a similar idea for non-linear mappings in a finite field. The Feistel round function of the 64-bit KN-cipher uses an affine mapping $e : \mathbb{F}_{2^{32}} \rightarrow \mathbb{F}_{2^{37}}$ to first transform the 32-bit input into a 37-bit value. After addition with a 37-bit round key, the resulting 37-bit value is then input to the non-linear permutation $g : x \rightarrow x^3$ in $\mathbb{F}_{2^{37}}$. Five bits of the output of g are then discarded to reduce the final output again to 32 bits. In summary, one application of the round function is given as

$$x_L||x_R \rightarrow x_R||x_L \oplus f(e(x_R) \oplus k_i)$$

where f consists of application g followed by discarding one bit. The KN cipher is a six-round Feistel design with six 37-bit independent round keys and is provably secure against differential attacks. However, it is vulnerable to an interpolation attack (see below) because of the low algebraic degree of the polynomial corresponding to the encryption function. The Feistel variant of our design — MiMC-$2n/n$ — can be easily recognized as a variant of the KN cipher, except for that we do not discard any bits (and hence always stay in the same field), add independent round constants and have a higher number of rounds. Indeed, our design more closely resembles $\mathcal{PURE}$, the cipher used in [JK97] to demonstrate the vulnerability of the KN cipher to interpolation attacks, except for the higher number of rounds in our design. The performance of both designs essentially differs linearly in by how much we extend the number of rounds. We note that our GCD attack in Section 4.2 also extends to $\mathcal{PURE}$ and allows to reduce the number of plaintext-ciphertext pairs required for a successful cryptanalysis.

3.2 The Pohlig-Hellman Cipher

The Pohlig-Hellman cipher was described in [PH78]. Choose a prime p. Pick $1 \leq k \leq p-2$ with $\gcd(k, p-1) = 1$ and $1 \leq d \leq p-2$ with $d = k^{-1} \bmod p-1$, with p public and k and d private. To encrypt the message $1 \leq m \leq p-1$ compute $c = m^k \bmod p$. To decrypt compute $m = c^d \bmod p$. Encryption and decryption take between $\log_2 p$ and $2\log_2 p$ multiplications depending on the Hamming weights of k and d. A key recovery attack solves the discrete logarithm problem in $\mathbb{F}_p$. The General Number Field Sieve solves this problem in complexity $\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln p)^{\frac{1}{3}}(\ln \ln p)^{\frac{2}{3}}\right) = L_p\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right]$. Thus for n-bit security, the number of multiplications required grows faster than $O(n)$.

3.3 Naor-Reingold PRF

The Naor-Reingold PRF [NR97] is a pseudorandom function whose security can be reduced to the decisional Diffie-Hellman problem. For a given $n \in \mathbb{N}$, primes p and q with q dividing $p-1$, an element $g \in \mathbb{F}_p^*$ of order q, and $n+1$ elements $a_0, \ldots, a_n \in \mathbb{Z}_q$, and an n-bit input $x_1, \ldots, x_n \in \mathbb{F}_2$ define

$$f_{p,q,g,\boldsymbol{a}}(x_1, \ldots, x_n) := g^{a_0 \prod_{x_i=1} a_i}$$

where $(g, \boldsymbol{a})$ is the secret key. Evaluation of the function corresponds to one exponentiation in $\mathbb{F}_p$ and n multiplications in $\mathbb{Z}_q$. Thus it takes between p and $2p$ multiplications in $\mathbb{F}_p$. As the security of this primitive can be reduced to the decisional Diffie-Hellman problem, just as with the Pohlig-Hellman cipher, for n bit security the number of multiplications grows faster than $O(n)$.

3.4 Ajtai, SWIFFT, SWIFFTX

SWIFFT [LMPR08] is a hash function family related to hard problems in lattices. In can be seen in the tradition of the work of Ajtai [Ajt96] and was used as a building block for the SWIFFTX SHA-3 submission [ADL+08]. The hash function consists of an application of the Number Theoretic Transform (NTT) over $\mathbb{Z}_{257}$ and in dimension 64 to $m = 16$ blocks of $n = 64$ bits. Each such transform costs $\frac{1}{2}n\log_2 n = 3 \cdot n = 192$ multiplications by a constant per 64 bits. The output of the NTT is then pointwise multiplied with 64 random fixed elements in $\mathbb{Z}_{257}$, costing another 64 multiplications. For $m \cdot n$ bits of input the algorithm scales linearly in m, so require $mn(1 + \frac{1}{2}\log_2 n)$ operations for $m \cdot n$ bits of input. On modern microprocessors most of these multiplications can be avoided by using precomputed lookup tables and some specifically chosen constants. However, it is not clear that these techniques translate to our setting. Furthermore, we note that multiplication by small constants can be more efficient than general multiplications in, e.g. homomorphic encryption schemes. On the other hand, the constants in an NTT are not small a priori. Still, our analysis might be somewhat pessimistic. We note that SWIFFT itself does not fulfil standard requirements for general purpose hash functions and that SWIFFTX addresses these issues

by running four SWIFFT instances (increasing the number of multiplications accordingly) and by introducing an S-box.

3.5 SPRING

SPRING [BBL+15] is a PRF proposal with security related to the Learning with Errors (LWE) problem. Similarly, to SWIFFT this construction employs an NTT over $\mathbb{Z}_{257}$, but at dimension $n = 128$. This costs $\frac{1}{2}n \log_2 n = 448$ multiplications in $\mathbb{Z}_{257}$. Additional, k multiplications in $\mathbb{Z}_{257}$ are required in a post-processing step for $k \in \{64, 128\}$ being the bit size of the input to the PRF. Hence, for $k = 128$ we expect 576 multiplications in $\mathbb{Z}_{257}$. We note that these multiplications can be realized efficiently on modern CPUs, but not necessarily in the scenarios targeted in this work.

3.6 Comparison

In Table 1 we compare MiMC with various block cipher and PRF designs. In Table 2 we compare MiMC with various cryptographic hash function proposals. In both cases, we notice a big difference between MiMC instantiations, and other designs for the two metrics that interest us: (1) the minimal number of multiplications needed to encrypt a block or at least n bits (minMULs), and (2) the number of multiplications per encrypted bit. For the GF(p) version of MiMC, the number of multiplications has to be multiplied by 2.

4 Design Rationale and Analysis of MiMC

In this section we explain the design rationale of the keyed permutation and argue its security. The monomial x^3 serves as the non-linear layer of the block cipher. Note that we can use x^3 to construct the cipher iff it is a permutation monomial in the field $\mathbb{F}_{2^n}$. The following well known result governs the choice of the monomial and size of the field in the design of MiMC.

Proposition 1 *Any monomial x^d is a permutation in the field $\mathbb{F}_{2^n}$ iff* $\gcd(d, 2^n - 1) = 1$.

Hence, $x \to x^3$ is not a permutation in $\mathbb{F}_{2^n}$ when n is even but only when n is odd. In particular, choosing thus $n = 2^t + 1$ ensures that x^3 is a permutation in $\mathbb{F}_{2^n}$.

Moreover, using the previous proposition, we can compute the inverse of the non-linear permutation x^3 in $\mathbb{F}_{2^n}$.

Lemma 1. *Let n an odd integer. The inverse of the non-linear function x^3 in $\mathbb{F}_{2^n}$ is given by x^s with $s := (2^{n+1} - 1)/3$.*

Proof. Given $y = x^3$, we are looking for an s such that $x = y^s$ in $GF(2^n)$, that is $x^{3 \cdot s} = x$. By Fermat's little theorem, this is equivalent to look for an s

Name	Security	minANDs	ANDs/bit	Remarks and Reference
AES-128	128	5120	40	GF(2) rep. [BP12] ([BMP13])
Simon	128	4352	34	[BSS+13]
Noekeon	128	2048	16	[DPVAR00]
Robin	128	3072	24	[GLSV14]
Fantomas	128	2112	16.5	[GLSV14]
LowMC	128	1132	8.85	[ARS+15]
Grain-128a	128	$4864 + 19 \cdot n$	19	[ÅHJM11]
Trivium	80	$1152 + 3 \cdot n$	3	[CP08]
Kreyvium	128	$1152 + 3 \cdot n$	3	[CCF+16]
		minMULs	MULs/bit	
AES-128	128	800	6.25	GF(2^8) rep. [CGP+12]
SPRING	128	576	4.5	[BBL+15]
Pohlig-Hellman	128	3072	≈ 1.5	[PH78,ENI13]
MiMC-129/129	129	**82**	**0.64**	this paper
MiMC-258/129	129	164	1.28	this paper

Table 1: Comparison of ciphers in encryption mode (excluding key schedule). We list the size-optimized variants. Note that in most cases multiplication refers to the field GF(2) (minANDs and ANDs/bit) whereas in MiMC and others multiplication is in a larger field(minMULs and MULs/bit). For stream ciphers we give the minANDs needed to generate n bits of output.

such that $3 \cdot s = 1 \pmod{2^n - 1}$. That is, there exists an integer t such that $3 \cdot s = 1 + t \cdot 2^n - 1$. By Proposition 1, we have that $\gcd(3, 2^x - 1) = 1$ if and only if x is odd (i.e. $\gcd(3, 2^x - 1) = 3$ if and only if x is even). For $t = 1$, we obtain $3 \cdot s = 2^n$ which is a contradiction. If t is equal to 2, then $3 \cdot s = 2^{n+1} - 1$. Since $n + 1$ is even (by hypothesis), then 3 divides $2^{n+1} - 1$. Finally, since x^3 is a permutation in $GF(2^n)$ for n odd (by previous proposition), then the inverse is unique and is given by $s := (2^{n+1} - 1)/3$. □

4.1 Computation Cost Model

In most models of computation field multiplication is considered to be more computationally expensive than addition. However, note that squaring is a linear operation in a binary field $\mathbb{F}_{2^n}$. Hence, if we consider the number of non-linear multiplications in a binary field then the number required to compute x^3 is one. In the SNARK setting, each witness variable (and possibly each constraint) is generated from a field operation more specifically from a field multiplication. As a consequence, computing x^3 generates two equations $x \cdot x = y$ and $y \cdot x = x^3$. Hence, in this setting we do not benefit from the linearity of squaring over the fields $\mathbb{F}_{2^n}$ and computing x^3 costs two multiplications. However, the cost of additions in these fields is still negligible compared to that of multiplication.

Name	Coll. Resist.	minANDs	ANDs/bit	Remarks and Reference
SHA-256	128	29000	56.64	[BCG+14])
SHA3-256	128	38400	35.29	[NIS14]
SHAKE128	128	38400	28.57	[NIS14]
		minMULs	MULs/bit	
SWIFFTX	112–256	16384	8.0	[ADL+08]
MiMCHash-256	129	**1293**	**2.52**	this paper
MiMCHash-256b	129	**971**	**1.89**	this paper

Table 2: Comparison of hash functions. We list the size-optimized variants. Note that in most cases multiplication refers to the field GF(2) (minANDs and ANDs/bit) whereas in MiMC multiplication is in a larger field (minMULs and MULs/bit).

Note that we can also disregard the cost of multiplication by a constant. Details on the form of equations involved in SNARK is given in Section 6.

We stress that although the cost of an addition is considered negligible compared to a multiplication, very large number of additions can reduce the efficiency of a design.

4.2 Security analysis

Our designs resist a variety of cryptanalysis techniques. The algebraic design principle of MiMC causes a natural concern about the security of the keyed permutation against algebraic cryptanalytic techniques. We describe several possible algebraic attacks (incl. a new "GCD" attack) against the design and analyze the resistance of the block cipher against these attacks. We also consider statistical attacks.

To summarize the following results, the number of rounds for the case of MiMC-n/n is derived from an interpolation attack, while the number of rounds for the case of MiMC-$2n/n$ is deduced from a Meet-in-the-Middle GCD attack.

Finally we discuss the case in which some restrictions are imposed (on data or memory requirements) for mounting an attack. We show that in this case it is possible to reduce the number of rounds.

Interpolation Attack. Interpolation attacks, introduced by Jakobsen and Knudsen [JK97], construct a polynomial corresponding to the encryption function without knowledge of the secret key. If an adversary can construct such a polynomial then for any given plaintext the corresponding cipher-text can be produced without knowledge of the secret key.

Let $E_k : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be an encryption function. For a randomly fixed key k, the polynomial $P(x)$ representing $E_k(x)$ can be constructed using Lagrange's

theorem, where x is the indeterminate corresponding to the plaintext. If the polynomial has degree d then we can find it using Lagrange's formula

$$P(x) = \sum_{i=1}^{d} y_i \prod_{1 \leq j \leq d, i \neq j} \frac{x - x_j}{x_i - x_j}$$

where $E_k(x_i) = y_i$ for $i = 1, 2, \ldots d$.

This method can be extended to a key recover attack. The attack proceeds by simply guessing the key of the final round, decrypting the cipher-texts and constructing the polynomial for $r - 1$ rounds. With one extra p/c pair, the attacker checks whether the polynomial is correct.

Observe that the number of unknown coefficients of the interpolation polynomial is $d + 1$ and that the complexity of constructing a Lagrangian interpolation polynomial is $\mathcal{O}(d \log d)$ [Sto85]. Hence, setting $d = 3^r$ with $r = r_{max} \approx n/\log_2(3)$ thwarts this attack. Note that no function mapping from $\mathrm{GF}(2^n)$ to $\mathrm{GF}(2^n)$ has degree $\geq 2^n$, since $T^{2^n-1} \equiv 1$ for each $T \in \mathbb{F}_{2^n}$ and the degree of the interpolation polynomial does not increase for $r > r_{max}$.

By the same argument, a similar result holds for the case of the Feistel network MiMC-$2n/n$. Indeed, at each round the left hand part of the state can be described as a polynomial of the left and of the right hand part of the plaintext, with at most $(3^r + 1) \cdot (3^{r-1} + 1) = 3^{2r-1} + 3^r + 3^{r-1} + 1$ unknown coefficients (observe that at round r, the degree of the polynomial is at most 3^r in the left part of the plaintext and 3^{r-1} in the right part). In a similar way, at each round the right hand part of the state can be described as a polynomial of the left and of the right hand part of the plaintext, with at most $(3^{r-1} + 1) \cdot (3^{r-2} + 1) = 3^{2r-3} + 3^{r-1} + 3^{r-2} + 1$ unknown coefficients Thus, the complexity of constructing this Lagrangian interpolation polynomial is approximately $\mathcal{O}(r \cdot 3^{2r-3})$, where a function mapping from $\mathrm{GF}(2^n)^2$ to $\mathrm{GF}(2^n)$ has degree at most 2^{2n}. With respect to MiMC-n/n, it follows that it is sufficient to choose $r = r_{max} + 2 \approx n/\log_2(3) + 2$ (i.e. two more rounds) to thwart this attack.

Moreover, note that in the chosen-plaintext scenario and in the case of MiMC-$2n/n$, an attacker can reduce the degree of the interpolation polynomial using several strategies. For example, for chosen plaintexts of the form $x||x^3$ the degree of the interpolation polynomial after r rounds is at most $2 \cdot 3^{r-1}$ in the left part of the plaintext and $2 \cdot 3^{r-2}$ in the right part, while for chosen plaintexts of the form $0||x$ the degree of the interpolation polynomial is at most 3^{r-1} in the left part of the plaintext and 3^{r-2} in the right part. Thus, for this second case, the interpolation polynomial of the right part of the text depends only by the right part of the plaintexts and has degree 3^{r-2}. In order to avoid the reduced degree of the polynomial, it is sufficient to add (at least) two rounds more to the number of rounds calculated for MiMC-n/n.

A meet-in-the-middle variant of the interpolation attack was also proposed in [JK97], constructing a polynomials $g(x) = h(y)$ instead of one polynomial $y = f(x)$. For MiMC-n/n, this approach does not produce an improvement due to the prohibitive degree of the inverse operation. In contrast, for MiMC-$2n/n$ we have that g and h may have degree $3^{r/2}$ in the left part of the plain-

text and $3^{r/2-1}$ in the right part only instead of degree 3^r and 3^{r-1} respectively. However, this lower degree comes at the price of increases computational cost. Indeed, constructing g and h requires solving a system of equation in $n = 2 \cdot (3^{r/2-1} + 1) \cdot (3^{r/2-2} + 1)$ unknowns costing $\mathcal{O}(n^\omega) = \mathcal{O}(3^{2 \cdot r-5})$ operations, where the hidden constant is ≥ 1 and we conservatively set the linear algebra constant $\omega = 2$. The chosen plaintext variant of this attack is quite similar. As before, the idea is to choose plaintexts in which the left part is fixed. In this way, one of the two interpolation polynomial depends only on one variable, the right part of the plaintext. Thus, constructing g and h requires solving a system of equation in $n = (3^{r/2-2} + 1) + (3^{r/2-1} + 1) \cdot (3^{r/2-2} + 1)$ unknowns costing $\mathcal{O}(n^2) = \mathcal{O}(3^{2 \cdot r-6})$ operations where the hidden constant is ≥ 1. In conclusion, with respect to MiMC-n/n and considering only the interpolation attack, it follows that it is sufficient to choose $r = r_{max} + 3 \approx n/\log_2(3) + 3$ (i.e. three more rounds) to thwart this attack. As we are going to show in the following, an higher number of rounds is requested to protect MiMC-$2n/n$ against the Meet-in-the-Middle GCD attack discussed in the next subsection.

We note that the complexity of an interpolation attack may decrease if the polynomial $P(x)$ is sparse for a chosen key. However, because we are adding random round constants in each round and x^3 is a permutation in $\mathbb{F}_{2^n}$ by construction, our $P(x)$ is not expected to be sparse[4].

Computing GCDs. From the description of MiMC, it is clear that factoring univariate polynomials recovers the key. However, if we are given more than one known plaintext-cipher-text pair, we can reduce the complexity further by computing a GCD of them. Denote by $E(k, x)$ the encryption of x under key k. For a pair $(x, y) \in \mathbb{F}_q^2$, $E(K, x) - y$ denotes a univariate polynomial in $\mathbb{F}_q[K]$ corresponding to (x, y). Note that in general, given plaintext/cipher text pair (x, y), it should be hard for a generic encryption scheme to compute the univariate polynomial $E(K, x) - y$ explicitly in the variable K (i.e. the secret key). However, this is not the case of MiMC, for which the polynomial $E(K, x) - y$ can be always computed explicitly, and it simply corresponds to the definition of encryption process (that is, the iterative application of the cubic function). Moreover, note that this attack may also be applied to $\mathcal{PURE}$, the cipher used

[4] This claim is supported by our experiments. In particular, for a field $\mathbb{F}_{2^n}$ and using x^3 as permutation, we observed:

- after 1 round, all terms appear (percentage: 100 %);
- after 2 round, 8 terms appear instead of 10 (percentage: 80 %);
- after 3 round, 19 terms appear instead of 28 (percentage: 67.86 %);
- after 4 round, 54 terms appear instead of 82 (percentage: 65.85 %);
- after 5 round, 161 terms appear instead of 244 (percentage: 66 %);
- after 6 round, 531 terms appear instead of 730 (percentage: 72.74 %);

and so on, where the percentage of the non-null terms continues to grow for the next rounds. For example, for the particular field $GF(2^{17})$, after 10 rounds almost all the terms are non-zero.

in [JK97] to demonstrate the vulnerability of the KN cipher to interpolation attacks, assuming round keys are not independent but linearly derived from k.

Consider now two such polynomials $E(K, x_1) - y_1$ and $E(K, x_2) - y_2$, with $y_1 = E(k, x_1)$ and $y_2 = E(k, x_2)$ for the fixed but unknown key k. It is clear that these polynomials share $(K - k)$ as a factor. Indeed, with high probability the greatest common divisor will be $(K - k)$. Thus, by computing the GCD of the two polynomials, we can find the value of k.

MiMC-n/n for a known plain text x corresponds to a polynomial having degree 3^r, where the leading monomial always has non-zero coefficient. Hence, we can recover k with a GCD computation of two polynomials at degree 3^r (indeed, considering differences of two polynomials $G(K, x_i) - y_i$ reduces this degree to $3^r - 1$ by canceling the leading term). It is well-known that the complexity for finding the GCD of two polynomials of degree d is $\mathcal{O}(d \log^2 d)$. Hence, the complexity of this attack is $\mathcal{O}(r^2 \cdot 3^r)$. For MiMC-$n/n$ the time complexity of this attack is higher than that of the interpolation attack.

More care must be taken for MiMC-$2n/n$. First of all, after r rounds the left hand part corresponds to a polynomial having degree 3^r while the right hand one corresponds to a polynomial having degree 3^{r-1} (in both cases the leading monomial always has non-zero coefficient). Moreover, in this case the meet-in-the-middle variant of this attack can be performed. That is, instead of constructing polynomials expressing ciphertexts as polynomials in the plaintext and the key, we can construct two polynomials $G'(K, x_i)$ and $G''(K, y_i)$ expressing the state in round $r/2$ as a polynomial in the key and the plaintext or ciphertext respectively. Then, considering $G'(K, x_1) - G''(K, y_1)$ and $G'(K, x_2) - G''(K, y_2)$ we can apply a GCD attack on polynomials of degree $3^{r/2-1}$, reducing the complexity to $\mathcal{O}(r^2 \cdot 3^{r/2-3})$ where the hidden constant is ≥ 1. Hence, the number of rounds must be increased to $r = 2 \cdot r_{max} \approx 2 \cdot n / \log_2(3)$ to thwart this attack[5]

Invariant Subfields. The algebraic structure of MiMC allows to mount a invariant subfield attack on the block cipher under a poor choice of round constants. That is, if all the round constants c_i and the key k are in subfield $\mathbb{F}_{2^m}$ of $\mathbb{F}_{2^n}$ then by choosing a plaintext $x \in \mathbb{F}_{2^m}$ an adversary can ensure that $E_k(x) \in \mathbb{F}_{2^m}$. This attack is thwarted by picking n to be prime. The only subfield is then $\mathbb{F}_2$ such that picking constants $\neq 1$ will be enough to avoid the attack.

[5] A more detailed explanation. By definition, the complexity of the GCD attack is given by $\mathcal{O}((r/2 - 1)^2 \cdot 3^{r/2-1})$. By simple calculation, observe that there exists a constant $C \geq 1$ (in particular $C \geq 9/4$) such that

$$\left(\frac{r}{2} - 1\right)^2 \cdot 3^{r/2-1} \leq C \cdot r^2 \cdot 3^{r/2-3}$$

for all $r \geq 1$. It follows that if a function $f(\cdot)$ is $f(r) = \mathcal{O}((r/2 - 1)^2 \cdot 3^{r/2-1})$ then it is also $f(r) = \mathcal{O}(r^2 \cdot 3^{r/2-3})$ with an hidden constant ≥ 1. Finally, it is simple to observe that $r^2 \cdot 3^{r/2-3} \geq 2^n$ for $r = 2 \cdot r_{max} \approx 2 \cdot n / \log_2(3)$.

Differential attacks. Differential cryptanalysis is one of the most widely used technique in symmetric-key cryptanalysis. The different types of cryptanalysis methods based on this technique depend on the propagation of an input difference through a given number of rounds of an iterative block cipher to yield a known output difference with high probability. The probability of the propagation often determines how many rounds can be attacked using this technique.

Given an input difference δ and an output difference δ', the differential probability of the round function is given as

$$\mathsf{Pr}\left(\delta \to \delta'\right) = |\{x \in \mathbb{F}_{2^n} : F(x+\delta) + F(x) = \delta'\}|/2^n \tag{1}$$

In our case the number of x satisfying $F(x+\delta)+F(x) = \delta'$ is determined by the non-linear function x^3. Hence it is enough to determine the size of the set

$$D = \{x \in \mathbb{F}_{2^n} : (x+\delta)^3 + x^3 = \delta', \delta \neq 0\}.$$

As this is a quadratic equation in x for any, there are at most two solutions to the equation. This implies $\mathsf{Pr}\left(\delta \to \delta'\right) \leq \frac{2}{2^n}$. This is sufficient to give any differential trail of at least two rounds a probability too low to be useful in an attack. A detailed analysis of the differential property of monomials of the form x^{2^t+1} in $\mathbb{F}_{2^n}$ can be found in [Nyb94] and in [Can97].

Linear attacks. Similar to differential attacks, linear attacks pose no threat to MiMC. Indeed, the cubic function is an *almost bent* or an *almost perfect nonlinear* (APN) function, i.e., differential 2-uniform, where an APN permutation provides the best resistance against linear and differential cryptanalysis. Thus, since its maximum square correlation is limited to 2^{-n+1} (cf. for example [AÅBL12] for details), any linear trail of the cubing function will have negligible potential after a few rounds.

Algebraic degree and higher-order differentials. As discussed above, the large number of rounds ensures that the algebraic degree of MiMC in its native field will be maximal or almost maximal. This naturally thwarts higher-order differential attacks when considering the difference as defined in the field (i.e., using the inverse of the field addition). But what happens to the degree when viewing the rounds as vectorial Boolean functions? As squaring is a linear operation in $\mathbb{F}_{2^n}$, it is also linear when viewed as vectorial function over $\mathbb{F}_2$. Cubing on the other hand introduces an additional multiplication which gives the round function an algebraic degree of 2 in every component when viewed as a vectorial Boolean function. Again, the large number of rounds should cause the degree to rise quickly and reach the limit of 2^n which is sufficient to thwart any higher-order differential attacks also when viewing the round function as a vectorial Boolean function.

Hash-specific security considerations. For usage of the MiMC permutation in the sponge mode as described in Section 2.3 we require the permutation to

not show non-trivial non-random behavior for up to 2^s input/output pairs. As specified in Section 2 the size of the permutation n determines the number of rounds (based on the GCD attack described above). As $2s < n$ for both MiMCHash-256 and MiMCHash-256b, this choices leaves us with an additional security margin, even if an hypothetical inside-out approach could double the number of rounds in an attack.

4.3 Additional security analysis for the case of restrictions on the complexity of the attack

For simplicity, in this section we limit our discussion to MiMC-n/n. However, it is straightforward to extend the following results to MiMC-$2n/n$ (where it is important to consider also the MitM variants of the attacks) or to the variants of MiMC over the prime fields.

In order to recover the number of rounds of MiMC-n/n (see Sect. 4), we did not consider any restrictions on the memory and/or on the number of pairs of plaintexts/ciphertexts that are available to the attacker. Here we show that if such restrictions hold, then it is possible to decrease the total number of rounds.

Restriction on the number of plaintext/ciphertext pairs available to the attacker. First we study the case in which only 2^m pairs of plaintexts/ciphertexts are available to the attacker, where $1 \leq m < n$.

This restriction is not negligible for the case of the interpolation attack. Indeed, if the interpolation polynomial has degree 3^r after r rounds (that is, if it has $3^r + 1$ coefficients), then the attacker needs at least $3^r + 2$ pairs to construct this polynomial (remember that she needs one pair to verify the guessed key). Thus, if $2^m - 1 < 3^{r-1} + 1$, the interpolation attack does not work since the attacker can not construct the interpolation polynomial. In other words, if 2^m pairs of plaintexts/ciphertexts are available to the attacker, then the minimum number of rounds r is equal to:

$$r = 1 + \left\lfloor \frac{\log_2(2^m - 2)}{\log_2 3} \right\rfloor \simeq 1 + \left\lfloor \frac{m}{\log_2 3} \right\rfloor.$$

Observe that this problem doesn't arises if the attacker has access to all possible pairs of plaintexts/ciphertexts. Indeed, remember that the maximum degree of the interpolation polynomial is $2^n - 2$ since $T^{2^n-1} \equiv 1$ for all $T \in \mathbb{F}_{2^n}$, that is the maximum number of coefficients of the interpolation polynomial is $2^n - 1$. Since the attacker works in $\mathbb{F}_{2^n}$ and she has access to all the possible pairs, then she can (theoretically) compute the interpolation polynomial.

Instead, note that the restriction on the number of texts available to the attacker has no influence on the GCD attacks, since two pairs are always sufficient to implement the attack. In this case, the cost of the attack remains $r^2 \cdot 3^r$ and we target a cost of 2^n so that $r^2 \cdot 3^r \simeq 2^n$, that is

$$r + 2\log_3(r) \simeq n \log_3(2).$$

Thus, we're interested to find a good approximation to the previous equivalence. Note that when the attacker has access to all the possible pairs of texts, then the interpolation attack is more efficient than the GCD attacks (see Sect. 4), thus we are not interested to study the GCD attack in detail (indeed, in this case if MiMC is secure against the interpolation attack then it is automatically secure against the GCD attack). Instead, when the attacker has access to a limit number of pairs, then we don't have any argumentation to do this claim a priori.

In order to find an approximated solution of the above equation, the idea is to use the mathematical methods provided by the *Perturbation Theory*. That is, the idea is to look for r of the form $r = n\log_3(2) - \varepsilon$, where $\varepsilon \ll n\log_3(2)$. By simple computation[6], it is simple to obtain:

$$r \simeq \left\lceil n\log_3(2) - 2\log_3\big(n\log_3 2\big) \right\rceil.$$

In conclusion, if only 2^m pairs are available (with $m < n$), then r is given by:

$$r = \max\left\{1 + \left\lfloor \frac{m}{\log_2 3} \right\rfloor, \left\lceil n\log_3(2) - 2\log_3\big(n\log_3 2\big) \right\rceil \right\}.$$

For the particular case in which $n = 129$, if $m \leq 115$ then $r = 74$, while $r = 1 + \lceil m \cdot \log_3 2 \rceil$ if $m > 115$.

Finally, we note that if an attacker has access to a limit number of texts, then the minimum number of rounds to protect MiMC against algebraic attacks is deduced using both the GCD attack and the interpolation attack. If instead sufficiently many pairs of plaintexts/ciphertexts are available to an attacker then the interpolation attack is sufficient to deduce the minimum number of rounds.

Restriction on the memory available to the attacker. Secondly, we consider the case in which there is a restriction on the memory available to the attacker. Note that this restriction affects the interpolation attack and the GCD attack in the same way. Indeed, the problem arises if the attacker is not able to store all the coefficients of the interpolation polynomial (and similar for the GCD attack). For example, in the case of MiMC-129/129 where the number of rounds is 82, the attacker needs $(3^{82} + 1) \cdot 129$ bits $\simeq 2^{134}$ bytes to store all the coefficients of the interpolation polynomial.

Thus, suppose that the attacker can store only 2^m coefficients, where $2^m \leq 2^n$. In MiMC-n/n, each coefficient requires of $n/4$ bytes to be stored. Thus, this is equivalent to suppose that only $n \cdot 2^{m-2}$ bytes are available to the attacker. Then, since the number of coefficients of the interpolation polynomial (similar for the GCD attack) depends on the number of rounds, we have the restriction that $2^m \leq 3^r + 1$, that is, the number of rounds r is given by:

$$r \simeq \lceil \log_3(2^m - 1) \rceil.$$

[6] We use the approximation $\log_3\big[\log_3(2^n - \varepsilon)\big] \approx \log_3\big[\log_3(2^n)\big]$.

For example, in the case of MiMC-129/129, suppose that only 2^{64} bytes (that is, approximately 84 exabytes or 84 million of terabytes) are available to the attacker. Then, 38 rounds are sufficient to protect MiMC-129/129 against the interpolation, the GCD and the other attacks. Time-memory trade-offs might well be possible, and we leave this as a topic for future research.

5 Variants

In this section, we discuss two variants of MiMC. One for instantiating MiMC over prime fields and one for extending the key size to increase security.

5.1 MiMC over prime fields

The above descriptions of MiMC can also be used to operate over prime fields i.e. a field $\mathbb{F}_p$ where p is prime. In that case, it needs to be assured that the cubing in the round function creates a permutation. For this, it is sufficient to require $\gcd(3, p-1) = 1$.

Following the notation as above, we can consider MiMC-p/p where the permutation monomial x^3 is defined over $\mathbb{F}_p$. The number of rounds for constructing the keyed permutation is $r = \left\lceil \frac{\log p)}{\log_2 3} \right\rceil$. In the Feistel mode, we define MiMC-$2p/p$ where the round function is defined over $\mathbb{F}_p$ and where the number of rounds is double with respect to MiMC-p/p. In both the constructions the r round constants are chosen as random elements in $\mathbb{F}_p$.

Our cryptanalysis from Section 4 transfers to this case except for the subfield attack which does not apply here.

5.2 Larger Keys

Instead of considering our simple iterative construction where we add the same key in each round, we may also consider the case where we have a key which is κ-times bigger than the block size n. In this case, we may consider an instance where we are cyclically adding κ independent keys to our rounds. Our i-th round function then becomes:

$$F_i(x) = (x \oplus k_{i \bmod \kappa} \oplus c_i)^3$$

It is clear that differential and linear cryptanalysis are not affected by this modification if we model MiMC as a Markov cipher. However, considering a larger key size does affect algebraic attacks. In particular, a simple GCD attack is not sufficient any more to recover the keys $k_0, k_1, \ldots, k_{\kappa-1}$. Instead, we may consider Resultants or Gröbner bases.

We consider the case where $\kappa = 2$. It is well-known [BKW93] that the maximum degree reached during a Gröbner basis computation of a bivariate system of equations is $\leq 2 \cdot \text{maxdeg}(P) + 1$, where $\text{maxdeg}(P)$ is the maximum degree of

our input system (i.e. 3^r in our case). Hence, from e.g. [BFS14], the complexity of solving such a system of equations is

$$\mathcal{O}\left(2 \cdot 3^r \cdot \binom{2 \cdot 3^r + 3}{2 \cdot 3^r + 1}\right).$$

Applying resultants, from [LMS13] we expect a complexity of

$$\tilde{\mathcal{O}}\left(d^{4.69}\right) = \tilde{\mathcal{O}}\left(3^{4.69\,r}\right).$$

Conservatively, we may anticipate a meet-in-the-middle attack which would reduce the cost of either of these attacks to a square root of the above estimates.

5.3 Different Round Functions

Considering the case $\mathrm{GF}(2^n)$, we may consider a round function of the form

$$F(x) = (x \oplus k \oplus c)^d$$

for generic exponents d. In particular, we have decided to limit our analysis to exponents of the form $2^t + 1$ and $2^t - 1$, for positive integer t (note that 3 is the only number that can be written in both ways). Remember that for MiMC-n/n, d has to satisfy the condition $\gcd(d, 2^n - 1) = 1$ in order to be a permutation, while in the case of MiMC-$2n/n$ (that is, for Feistel Networks) this condition is not necessary.
For further analysis, we recall the Lucas's Theorem:

Theorem 1. *For non-negative integers m and n and a prime p, the following congruence relation holds:*

$$\binom{m}{n} \equiv \prod_{i=0}^{k} \binom{m_i}{n_i} \pmod{p},$$

where $m = m_k p^k + m_{k-1} p^{k-1} + ... + m_1 p + m_0$ and $n = n_k p^k + n_{k-1} p^{k-1} + ... + n_1 p + n_0$ are the base p expansions of m and n respectively, using the convention that $\binom{m}{n} = 0$ if $m < n$.

Exponents of the form $2^t + 1$ (with $t > 1$) have the nice property that the cost to compute x^{2^t+1} does not depend on t, i.e. it requires only one multiplication (in some applications). Moreover, the degree of the resulting r-round interpolation polynomial is $(2^t + 1)^r$, which is significantly higher than 3^r even for "small" t. The major problem of this kind of exponents is that the corresponding interpolation polynomials are in general sparse. For example, using Lucas's Theorem, it is very easy to note that just after one round the polynomial has only 4 terms instead of $2^t + 2$:

$$\begin{aligned}(x \oplus k)^{2^t+1} &\equiv_2 (x \oplus k)^{2^t} \cdot (x \oplus k) \equiv_2 \\ \equiv_2 (x^{2^t} \oplus k^{2^t}) \cdot (x \oplus k) &\equiv_2 x^{2^t+1} \oplus k \cdot x^{2^t} \oplus k^{2^t} \cdot x \oplus k^{2^t+1}.\end{aligned}$$

Using the same technique, after r rounds, the number of terms of the polynomial is upper bounded by $3^r + 1$, which is (much) smaller than $(2^t + 1)^r + 1$. Note that $3^r + 1$ is exact the same upper bounded obtained for the exponent 3 (which corresponds to $t = 1$). Thus, the number of rounds to guarantee the security against the algebraic attacks doesn't change choosing exponent of the form $2^t + 1$ for $t > 1$. That is, both from the security point of view and from the implementation one, there is no advantage to choose exponents of the form $2^t + 1$ greater than 3.
Similar considerations can be done also for exponents of the form $2^t + 2^s = 2^s \cdot (2^{t-s} + 1)$, where $s < t$.

For this reason, coefficients of the form $2^t - 1$ are more interesting. Indeed, in this case it is very easy to prove that the interpolation polynomial is not sparse:

$$(x \oplus k)^{2^t-1} \equiv_2 \bigoplus_{i=0}^{2^t-1} x^i \cdot k^{2^t-1-i},$$

since

$$\binom{2^t - 1}{i} \equiv_2 1 \qquad \forall i \in \{0, 1, \ldots, 2^t - 1\}.$$

On the other hand, in order to compute x^{2^t-1}, we need more multiplications and square operations. Thus, a natural question is if it is possible to minimize the total number of multiplications necessary to compute the ciphertext choosing an exponent of the form $2^t - 1$ different from 3.

There are different ways to compute g^e where $g \in \mathbb{F}_{2^n}$ and $e = 2^t - 1$ for some $t \geq 2$, the classical algorithm being the square-and-multiply algorithm, cf. [MVO96, Sect. 14.6]. For this algorithm, the number of multiplications requested for this exponent is equal to the number of squares $t-1$. In Algorithm 1, we give a slight variation of the original algorithm.

Data: $g \in \mathbb{F}_{2^n}$ and $e = 2^t - 1$ for some $t \geq 2$
Result: g^e
$g_0 \leftarrow g$;
$g_1 \leftarrow g^2 \cdot g$;
$A \leftarrow 1$;
for i from *0 to* $\lfloor t/2 \rfloor$ **do**
  $A \leftarrow (A^2)^2$;
  $A \leftarrow A \cdot g_1$;
end
if $t \bmod 2 \neq 0$ **then**
  $A \leftarrow A^2$;
  $A \leftarrow A \cdot g_0$;
end
return A.

Algorithm 1: Modular exponentiation with cache

By simple computation, the number of multiplications for the previous algorithm is $\lceil t/2 \rceil$, while the number of squares is $t-1$. Observe that with respect to the original algorithm, it requires precomputation and to store the quantity $g^2 \cdot g$. Thus, for our purpose, this algorithm is better than the original one (for the case $e = 2^t - 1$). This algorithm can be improved[7], but for our purpose it suffices.

Thus, using the previous analysis about the number of rounds, the total number of multiplications m and of squares s for MiMC-n/n (analogous for MiMC-$2n/n$) is

$$m = \left\lceil \frac{t}{2} \right\rceil \cdot \left\lceil \frac{n}{\log_2(2^t - 1)} \right\rceil \qquad s = (t-1) \cdot \left\lceil \frac{n}{\log_2(2^t - 1)} \right\rceil.$$

For example, for $n = 129$, the best result is obtained for $t = 4$ (that is for the exponent 15)[8], for which the total number of multiplications is 66 (instead of 82 for the exponent 3), while the number of squares is 99 (instead of 82 for the exponent 3).

Note that the sum of the total number of multiplications m and of the total number of squares s is almost constant for each choice of t.

Finally, only for completeness, it is also possible to extend the previous analysis to the case $GF(p)$. In this case, since the square operation is not linear, it counts as a multiplication. Thus, if we consider an exponent of the form $2^t - 1$, the total number of multiplications m for MiMC-p/p is

$$m = \left(\left\lceil \frac{t}{2} \right\rceil + t - 1 \right) \cdot \frac{\log(p-1)}{\log(2^t - 1)}.$$

To conclude, if the cost of a square operation is negligible with respect to the cost of a multiplication (that is, if the square operation is linear), then it is possible to minimize the total number of multiplications choosing an exponent of the form $2^t - 1$ different from 3. Instead, when the number of square operations can not be ignored (as for example in the case of SNARK settings or in the $GF(p)$ case), the choice of an exponent of the form $2^t - 1$ different from 3 does not offer any advantage due to the fact that the number $m+s$ is almost constant.

6 Application and Implementation

We implemented the MiMC block cipher and hash function in C++ using NTL [Sho]. Note that we put no restriction on the irreducible polynomial to represent the finite field $\mathbb{F}_{2^n}$ in our proposal.

[7] For example, suppose that $t \geq 8$. The idea is to precompute g_0, g_1 (defined as before) and also $g_2 := (g_1)^4 \cdot g_1$. Thus, in the *for* loop $0 \leq i \leq \lfloor t/4 \rfloor$ and $A \leftarrow A^8 \cdot g_2$. Finally, after the *for* loop and before the *if*-statement, one has to take care of the case $t \mod 4 \neq 0$.

[8] Actually, the best result is obtained for $t = 6$, that is for the exponent 63. But since $\gcd(63, 2^{129} - 1) = 7$, the round function defined using the exponent 63 is not a permutation.

6.1 Verifiable Computation and SNARK

Recently, several techniques have been proposed to achieve practical or nearly practical verifiable computation through constructions such as Pinocchio [PHGR16] and zk-SNARK. A special kind of *Succinct Non-interactive Argument of Knowledge* or SNARK was proposed in 2014 to build Zerocash [BCG$^+$14] — a digital currency similar to Bitcoin but achieving anonymity. In [BSCG$^+$13] an implementation of a publicly verifiable non-interactive argument system is given.

The main idea of the SNARK is to provide a circuit whose satisfiability enables a verifier to check correctness of an underlying computation. In this concrete implementation, we focus on the (zk)SNARK for arithmetic circuit satisfiability. The main target of our design proposals is to improve the efficiency of (zk)SNARK when they are used as cryptographic primitives in a SNARK setting.

An $\mathbb{F}$-arithmetic circuit takes input from the field $\mathbb{F}$ and its gates produce output in $\mathbb{F}$. Also the circuits considered here consist of bilinear gates only. Arithmetic circuit satisfiability (ACS) is defined as follows:

Definition 1. *The ACS problem of an $\mathbb{F}$-arithmetic circuit $\mathcal{C} : \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^l$ is depicted by the relation $\mathcal{R} = \{(x, a) \in \mathbb{F}^n \times \mathbb{F}^h : \mathcal{C}(x, a) = 0^l\}$ such that its language is $L = \{x \in \mathbb{F}^n : \exists a \in \mathbb{F}^h \text{ s.t } \mathcal{C}(x, a) = 0^l\}$.*

Since the circuit consists of bilinear gates only, we aim to minimize the number of NLM or field multiplications in our design. The addition in the field, which is the same as bitwise XOR, is a comparatively less expensive operation. The SNARK algorithm generates the proof for satisfiability of a system of *rank-1 quadratic constraints* over a finite field. This system of constraints is defined as below.

Definition 2. *A system of rank-1 quadratic equations over a field $\mathbb{F}$ is a sequence of tuples $((A_i, B_i, C_i), n)$ for $i = 1, \ldots, N_c$ and $A_i, B_i, C_i \in \mathbb{F}^{1+N'}$ such that $n \leq N'$. This system is satisfiable with an input $x \in \mathbb{F}^n$ if there is a witness $w \in \mathbb{F}^{N'}$ such that*

$$\langle A_i, w\rangle \cdot \langle B_i, w\rangle = \langle C_i, w\rangle \quad \forall i = 1, \ldots, N_c$$

Here N_c is the number of constraints and N' is the number of variables.

The number of such constraints contributes to the efficiency of the SNARK algorithm. From the above definition it is also clear that in a SNARK setting over $\mathbb{F}_{2^m}$ we can not ignore the squaring as linear operation.

MiMC in the SNARK setting. In MiMC, each round can be expressed with the following equations

$$X + \underbrace{k_i + C_i}_{\alpha} + U = 0 \tag{2}$$

$$U \cdot U = Y \tag{3}$$

$$Y \cdot U = Z \tag{4}$$

where k_i, C_i are the round key and constants respectively. Note that the above 3 equations can be combined to form one rank-1 quadratic constraint (as in definition 2)

$$(X + \alpha)(X + \alpha + Y) = Y + Z \tag{5}$$

For the MiMCHash the round key is fixed to a constant hence α can be treated as a constant in this equation. Note that the number of witness per round of MiMC is 2. Therefore the total number of witness for the fixed key permutation is $2 \cdot R$, where $R \approx \frac{n}{\log 3}$ is the number of rounds and n is the block size. The witness generation requires one constant addition (XOR) and two multiplications in the corresponding field. The complexity of the prover algorithm of SNARK (appendix E in [BSCG+13]) is dominated by $O(N_c \log N_c)$ where N_c is the number of rank-1 constraints.

LowMC in the SNARK setting. In LowMC, each round consists of Sbox (3-bit), matrix multiplication (over $\mathbb{F}_2$), round key and constant addition (XOR). Each 3-bit Sbox application can be written as

$$b \cdot c = a + z_1 \tag{6}$$

$$a \cdot (c + 1) = b + z_2 \tag{7}$$

$$a \cdot (b + 1) = b + c + z_3 \tag{8}$$

The above three equations can be combined to form 2 rank-1 constraints as following

$$b \cdot c = a + z_1 \tag{9}$$

$$a \cdot (b + c) = c + z_2 + z_3 \tag{10}$$

The witness generation for each Sbox requires 3 multiplications and 6 additions (out of which 2 are constant additions) over $\mathbb{F}_2$. In each round there are m Sboxes. Hence per round the witness generation process will require $3m$ multiplications and $6m$ ($2m$ of them are constant addition) additions per round. Suppose N_b is the block size of the permutation. Then there will be approximately $(l-1) \cdot N_b$ additions over $\mathbb{F}_2$ due to linear layer of LowMC in each round, where l is the average number of non-zero entries in each row of the random matrix of the linear layer. Also there will be N_b constant additions over $\mathbb{F}_2$ which is due to round constant and key addition. The total number of rank-1 constraints for R rounds of LowMC will be $R \cdot 2m$.

Note that the number of additions are much more in comparison with the number of multiplication over $\mathbb{F}_2$.

Remark 1. For the MiMC permutation, the operations are performed over a larger field e.g $\mathbb{F}_{2^{1025}}$. Indeed the cost of a single multiplication is higher in the larger field compared to a multiplication over $\mathbb{F}_2$. Moreover, the number of additions are significantly more than the number of multiplications (see Table 3). Although in the cost model the cost of addition is much less than the cost of multiplication, very large number of additions over $\mathbb{F}_2$ brings down the efficiency of LowMC in SNARK setting in comparison to MiMC. On the other hand, in MiMC the number of additions per round is one.

Experimental results. Following the `libsnark` [Lab] implementation we have implemented a prototype of SNARK for generating the circuit and witness for MiMC permutation for different block sizes and MiMCHash-256. One important target application of MiMC is SNARK or SNARK like algorithms. We have measured the time taken by MiMCHash for processing a single block and compared it with the time taken by SHA-256 using the `libsnark` implementation.

For processing a single block i.e. for hashing a single block message our MiMC implementation in the SNARK setting requires ≈ 7.8 milliseconds to generate the arithmetic circuit and witness while SHA-256 takes ≈ 73 milliseconds.

Since LowMC was designed for MPC/ZK applications we have also implemented it in the SNARK setting. A comparison of LowMC with MiMC is given in Table 3.

	MiMC	LowMC		Keccak-$[1600, 24]$
		$R = 16$ $m = 196$	$R = 55$ $m = 20$	
total time	7.8ms	90.3ms	271.2ms	75.8ms
constraint generation	6.3ms	13.5ms	9.2ms	65.2ms
witness generation	1.5ms	76.8ms	262.0ms	10.6ms
# addition	646	8420888	28894643	422400
# multiplication	1293	9408	3300	38400
# rank-1 constraint	646	4704	2200	38400

Table 3: Comparison of LowMC and MiMC with block size 1025 and the corresponding parameters for LowMC and Keccak permutation with specified parameters. For all implementations we have used the `-O3` optimization option of the gcc compiler. For LowMC, the number of rounds and the number of Sboxes per round are denoted as R and m respectively.

If we intend to use the LowMC permutation to construct a hash function using Sponge mode then the block size of LowMC should be 1025 bit for achieving

the same security level as SHA-256 or MiMCHash-256. We have implemented LowMC with the updated parameter-set v2 from[ARS+16] with this block size and two possible choices for the parameters (R, m), where R and m are number of rounds and number of Sbox per round respectively. One is minimizing the number of rounds for the given block size and security requirements, the other one is minimizing the number of ANDs/bit. Both are derived from the round formula given in [ARS+16]. LowMC is mainly a block cipher and the original proposal did not provide any suggestion to construct a secure hash function using the permutation. However if used in the sponge mode then the performance of the resulting hash function can be approximated by the performance of the LowMC permutation in SNARK setting.

We have also compared the performance of the Keccak-$[1600, 24]$ [NIS14] permutation when used for the SHA-3 and SHAKE hash function in our SNARK setting. Note that the truncation after a Keccak permutation can be expressed as equality constraints. In fact the performance for the SHAKE128 or SHA3 are almost same as the Keccak-$[1600, 24]$. The performance comparison in the Table 3 shows that MiMC is significantly more efficient than LowMC and SHA-3 in SNARK setting.

For all field operations we have used the NTL library together with the `gf2x` library. All computations were carried out on an Intel Core i7 2.10GHz processor with 16GB memory and we took the average over ≈ 2000 repetitions. As a design with an unusual imbalance between ANDs and XORs, the comparison with LowMC variants is interesting as it gives an example where the number multiplications alone can no longer be used as a hint for the eventual performance. Where the round-minimized LowMC variant is more than 10 times slower with about 8 times more multiplications, reducing the number of ANDs in the other LowMC variant at the expense of many more rounds does not have the expected effect: The runtime grows again. The reason is the huge amount of XOR computations whose cost is clearly are no longer negligible. This shows the limits of a simplified metric that focuses on AND gates (or multiplication gates) also.

All implementations in C++ can be found on `https://github.com/byt3bit/mimc_snark.git`.

6.2 Direct implementation

For the sake of completeness we provide a brief discussion of the complexity for the direct implementation MiMC, but stress that it has limited impact on the performance on our target platforms. Each round of MiMC-n/n performs one multiplication in the field $\mathbb{F}_{2^n}$. For the considered values of n this computation of x^3 becomes computationally expensive, since it is not feasible to use the efficient lookup table method even for $n = 32, 64$.

The evaluation of x^3 can be reduced to field multiplication. Since the problem is frequently encountered in many public-key cryptographic algorithms and protocols, efficient field multiplication is a well studied area in the literature. One strategy for efficient field multiplication is to use lookup tables. Indeed, several algorithms [GP97,DWBV+96,HMV93] are proposed in the literature which

use precomputed lookup tables to improve the efficiency of finite field multiplication. We briefly describe the complexity for evaluating the monomial using several algorithms from the literature.

	Number of Instructions		Look-up Table	
	XOR	ADD,SUB, SHIFT, AND	Bit size	No. of Access
[HMV93]	$2g^2$	$g^2\left(\frac{3}{2} - \frac{1}{2(2^b-1)}\right)$	$2b2^b$	$3g^2$
[GP97]	$6g^{\log 3} - 8 \cdot g + 2$	$g^{\log 3}$	$2b2^b$	$3g^{\log 3}$
[KA98]	$4g^2$	—	$(2b-1)2^{2b}$	$2g^2 + g$
[Has00]	$\left(\frac{1}{2}(g+1)(b+3) - 4\right)\lceil\frac{n}{w}\rceil$	$(g-1)\lceil\frac{n}{w}\rceil + 4g - 2$	$(b+d)2^b$	$(g-1)\lceil\frac{b+d}{w}\rceil$

Table 4: Complexities of different algorithms for implementing field multiplications

In all lookup-table based multiplication algorithms above, b is the size of the internal data path of the processor. Any element in $\mathbb{F}_{2^n}$ is partitioned as a collection into g groups each having b bits. If n is not a multiple of b then the most significant group will contain $n \pmod b$ bits. Note that the algorithm in [HMV93] requires n to be multiple of b. Furthermore, d denotes the degree of the second highest monomial (with non-zero coefficient) in the irreducible polynomial that defines the field $\mathbb{F}_{2^n}$ and w denotes the word size of processor. The resources of a processor are optimally utilized when $b = w$. For example in a 32 bit processor two polynomials can be added using $\lceil\frac{n}{32}\rceil$ XOR instructions. However choosing $b = w$ in this case increases the size of the lookup table to 2^5 GB for the algorithms from [HMV93,GP97]. On the other hand choosing $b < w$ may imply lower utilization of processor's resources. The algorithm described in [Has00] proposes a better utilization of resources when a small value of b is chosen to keep the size of the lookup table sufficiently small. Also, this algorithm does not require n to be multiple of b.

6.3 Generic masking against side-channel attack

Side-channel attacks exploit different types of physical leakage of information e.g. power consumption or EM emanations during the execution of cryptographic algorithms on a device for recovering sensitive variables (e.g. secret key). Masking is a well known technique to prevent implementations of cryptographic algorithms from such attacks. Most of the masking schemes usually protect an implementation against first-order attacks. Over the past years several higher-order side-channel attacks were proposed and demonstrated successfully against many well-known cryptographic algorithms. Higher order masking schemes are useful to protect a cryptographic algorithm against such attacks.

In a higher order masking scheme a sensitive variable (e.g. variables involving secret keys) is split into $t+1$ shares where t is known as the order of masking. It

has been shown that the complexity of side-channel attacks increases exponentially with the masking order.

In FSE 2012 a generic higher order masking scheme [CGP+12] was proposed by Carlet, Goubin, Prouff, Quisquater and Rivain. For masking an S-box using CGPQR scheme we need to consider the polynomial corresponding to the S-box, which can be easily computed from the S-box table using Lagrange's theorem in a field $\mathbb{F}_{2^n}$. In CGPQR masking scheme evaluation of this polynomial is protected against higher order attacks. For example, let x be a secret variable for which we evaluate a function $f(x)$. Let $x_0, x_1, \ldots, x_t$ are the $t+1$ shares corresponding to this variable such that $x = \bigoplus_{i=0}^{t} x_i$. Any linear function $\ell(x)$ is easy to mask since $\ell(x) = \ell(x_0) \oplus \ldots \oplus \ell(x_t)$. However masking a non-linear function is not as easy as linear or affine functions.

The operations necessary for evaluating a polynomial in $\mathbb{F}_{2^n}$ are addition, multiplication by a scalar, squaring and regular multiplication. For tth order masking any affine and linear operation in $\mathbb{F}_{2^n}$ requires $\mathcal{O}(t)$ logical operations, whereas regular multiplication requires $\mathcal{O}(t^2)$ logical operations. Hence regular multiplication is significant operation in CGPQR masking scheme and its efficiency can be increased by minimizing the number of regular multiplications in a field for a cryptographic algorithm.

MiMC is constructed using a monomial x^3 in $\mathbb{F}_{2^n}$. Evaluation of this monomial in each round requires only one multiplication and hence is optimized for CGPQR higher order masking scheme.

7 Conclusions

We have reconsidered a 20-year old cipher design idea, given a thorough security analysis, and demonstrated that it can be very competitive in emerging new applications of symmetric cryptography: SNARKs. It might seem that the usefulness of the design is limited to this setting, as the number of rounds is high compared to other more "traditional" designs for symmetric primitives. However there is evidence that the opposite is true, which was recently discovered in a follow-up work [GRR+16]. Due to its very simple design and despite the high number of rounds, it also turned out to be very competitive in a very different application setting: The currently fastest known MPC protocols with security against active adversaries. This clearly shows that there is a good use-case for designs which work natively in GF(p), and we hope that MiMC can inspire more design and cryptanalysis in this direction.

Acknowledgements We thank Alessandro Chiesa, Eran Tromer and Madars Virza for helpful discussions on SNARKs. The work in this paper has been partially supported by the Austrian Science Fund (project P26494-N15) and by the EU H2020 project Prismacloud (grant agreement nr. 644962). Albrecht was supported by EPSRC grant EP/L018543/1 "Multilinear Maps in Cryptography".

References

[AÅBL12] Mohamed Ahmed Abdelraheem, Martin Ågren, Peter Beelen, and Gregor Leander. On the distribution of linear biases: Three instructive examples. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 50–67. Springer, Heidelberg, August 2012.

[ADL+08] Yuriy Arbitman, Gil Dogon, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swifftx: A proposal for the sha-3 standard. Submission to NIST, 2008.

[ÅHJM11] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of grain-128 with optional authentication. *IJWMC*, 5(1):48–59, 2011.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.

[ARS+16] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. Cryptology ePrint Archive, Report 2016, 2016. `http://eprint.iacr.org/`.

[BBL+15] Abhishek Banerjee, Hai Brenner, Gaëtan Leurent, Chris Peikert, and Alon Rosen. SPRING: Fast pseudorandom functions from rounded ring products. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 38–57. Springer, Heidelberg, March 2015.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE Computer Society, 2014.

[BDPA08] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[BFS14] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the Complexity of the F5 Gröbner basis Algorithm. *Journal of Symbolic Computation*, pages 1–24, September 2014. 24 pages.

[BKW93] T. Becker, H. Kredel, and V. Weispfenning. *Gröbner bases: a computational approach to commutative algebra*. Springer-Verlag, 1993.

[BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *Journal of Cryptology*, 26(2):280–312, 2013.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES S-box. In *Information Security and Privacy Conference (SEC)*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors,

CRYPTO 2013, Part II, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. `http://eprint.iacr.org/2013/404`.

[Can97] Anne Canteaut. Differential cryptanalysis of Feistel ciphers and differentially δ-uniform mappings. In *Workshop on Selected Areas in Cryptography, SAC '97, Workshop Record*, pages 172–184, 1997.

[CCF+16] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 313–333. Springer, 2016.

[CFH+15] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015*, pages 253–270. IEEE Computer Society, 2015.

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for s-boxes. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.

[CP08] Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008.

[DPVAR00] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. Nessie proposal: Noekeon. In *First Open NESSIE Workshop*, 2000.

[DWBV+96] Erik De Win, Antoon Bosselaers, Servaas Vandenberghe, Peter De Gersem, and Joos Vandewalle. A fast software implementation for arithmetic operations in gf(2n). In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 65–76. Springer Berlin Heidelberg, 1996.

[ENI13] ENISA. Algorithms, key sizes and parameters report – 2013 recommendations. Technical report, European Union Agency for Network and Information Security, October 2013.

[GLSV14] Vicente Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-designs: Bitslice encryption for efficient masked software implementations. In *Fast Software Encryption (FSE)*, LNCS. Springer, 2014. To appear.

[GP97] Jorge Guajardo and Christof Paar. Efficient algorithms for elliptic curve cryptosystems. In Jr. Kaliski, BurtonS., editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 342–356. Springer Berlin Heidelberg, 1997.

[GRR+16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. MPC-friendly symmetric key primitives. In Edgar R.

Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 430–443. ACM, 2016.

[Has00] M. Anwarul Hasan. Look-up table-based large finite field multiplication in memory constrained cryptosystems. *IEEE Trans. Comput.*, 49(7):749–758, July 2000.

[HMV93] Greg Harper, Alfred Menezes, and Scott Vanstone. Public-key cryptosystems with very small key lengths. In RainerA. Rueppel, editor, *Advances in Cryptology — EUROCRYPT' 92*, volume 658 of *Lecture Notes in Computer Science*, pages 163–173. Springer Berlin Heidelberg, 1993.

[JK97] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In Eli Biham, editor, *Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer Berlin Heidelberg, 1997.

[KA98] Cetin K. Koc and Tolga Acar. Montgomery Multiplication in GF(2k). *Designs, Codes and Cryptography*, 14(1):57–69, 1998.

[KN95] Lars R. Knudsen and Kaisa Nyberg. Provable security against a differential attack. *Journal of Cryptology*, 8(1):27–37, 1995.

[KR11] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011.

[Lab] SCIPR Lab. `libsnark`. `https://github.com/scipr-lab/libsnark`.

[LMPR08] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 54–72. Springer, Heidelberg, February 2008.

[LMS13] Romain Lebreton, Esmaeil Mehrabi, and Éric Schost. On the complexity of solving bivariate systems: the case of non-singular solutions. In Manuel Kauers, editor, *International Symposium on Symbolic and Algebraic Computation, ISSAC'13, Boston, MA, USA, June 26-29, 2013*, pages 251–258. ACM, 2013.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016*, volume 9665 of *Lecture Notes in Computer Science*, pages 311–343. Springer, 2016.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[NIS14] NIST. DRAFT FIPS PUB 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, 2014.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97*, pages 458–467. IEEE Computer Society, 1997.

[Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 55–64. Springer Berlin Heidelberg, 1994.

[PH78] Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[PHGR16] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: nearly practical verifiable computation. *Commun. ACM*, 59(2):103–112, 2016.

[Sho] V. Shoup. Number theory library 5.5.2 (ntl) for c++. `http://www.shoup.net/ntl/`.

[Sto85] H-J Stoss. The complexity of evaluating interpolation polynomials. *Theoretical computer science*, 41:319–323, 1985.

A SNARK prover algorithm

Here we give a brief description of the parameters chosen to implement the prover algorithm for MiMCHash-256 using the MiMC-1025/1025 permutation with a fixed key. We also briefly describe a part the prover algorithm for MiMC in a SNARK setting which requires polynomial interpolation. For a more detailed description of the SNARK algorithm we refer the readers to [BSCG$^+$13].

A.1 Complexity of the prover algorithm

Let S be the system of rank-1 quadratic constraints as described in Definition 2 of the article with the tuples $(A_i, B_i, C_i) \in \mathbb{F}^{N'+1}$ for $i \in [N]$. Fix an arbitrary subset $\mathcal{X} = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ of $\mathbb{F}$ such that $\alpha_i = \omega^{i-1}$ for $i \in [N]$ and ω is the N th root of unity. Given an input $x \in \mathbb{F}^m$ and witness $w \in \mathbb{F}^{N'}$ such that $(x, w) \in \mathcal{R}$. The prover algorithm performs the following steps :

1. Choose $\delta_1, \delta_2, \delta_3$ independently at random from the field $\mathbb{F}$
2. Construct the polynomial

$$Q(z) := \frac{F(z)G(z) - H(z)}{U(z)}$$

where $U(z) := z^N - 1$ and F, G, H are univariate polynomials of degree N defined as

$$F(z) = \underbrace{F_0(z) + \sum_{i=1}^{N'} w_i F_i(z)}_{F'(z)} + \delta_1 U(z)$$

$$G(z) = G_0(z) + \sum_{i=1}^{N'} w_i G_i(z) + \delta_2 U(z)$$

$$H(z) = H_0(z) + \sum_{i=1}^{N'} w_i H_i(z) + \delta_3 U(z)$$

Here $F_i, G_i, H_i : \mathcal{X} \to \mathbb{F}$ are the Lagrange basis functions for the corresponding polynomials satisfying the following conditions

$$F_i(\alpha_j) = A_j(i), G_i(\alpha_j) = B_j(i), H_i(\alpha_j) = C_j(i)$$

for each $i \in \{0, 1, \ldots, N'\}$ and $j \in [N]$. Note that for any input x and witness w if $(x, w) \in \mathcal{R}$ then $U(z)$ divides $F(z)G(z) - H(z)$.

3. Output the vector $(1, \delta_1, \delta_2, \delta_3, w, q)$ such that $q = (q_0, q_1, \ldots q_N)$ represents the polynomial Q.

Note that each of the polynomials F', G', H' (hence F, G, H) can be computed using an inverse FFT which has a complexity $O(N \log N)$. Next a multiplicative coset $\mathcal{Y} := \gamma\mathcal{X}$ of $\mathcal{X} = \{\alpha_1, \ldots \alpha_N\}$ is chosen such that $\gamma \in \mathbb{F} - \mathcal{X}$. The polynomial $Q(z)$ is computed in two steps

- Evaluate $Q'(z) := \frac{F'(z)G'(z) - H'(z)}{U(z)}$ on $\mathcal{Y}$ point-by-point using the evaluations of F', G', H', U on $\mathcal{Y}$
- Compute $Q'(z)$ using inverse FFT and compute $Q(z) := Q'(z) + \delta_2 F'(z) + \delta_2 G'(z) + \delta_1\delta_2 U(z) - \delta_3$.

The first step out of the above two takes $O(N)$ field operations and the inverse FFT has the complexity $O(N \log N)$.

A.2 Parameters for MiMCHash-256

Over $\mathbb{F}_{2^n}$ We describe the parameter choices for $n = 1025$. The hash function constructed over this particular field promises the same level of security as SHA-256. For processing a single block we use the MiMC-1025/1025 over $\mathbb{F}_{2^{1025}}$. The two constraints in each round of MiMC permutation can be combined to obtain a single rank one quadratic constraint. Hence we get approximately $1025/\log(3) \approx 646$ constraints from the permutation together plus an additional constraint for compression function making the total number of constraints 647. Note that each round introduces two variables in the constraints hence the number of witness is 1293 where $w_1 = x \in \mathbb{F}_{2^{1025}}$ is the input to the hash function and $w \in (\mathbb{F}_{2^{1025}})^{1293}$.

In the prover algorithm the number of constraints N should be such that the principal N-th root exists in $\mathbb{F}_{2^{1025}}$. To satisfy this condition we choose $N = 1801$ (since 1801 divides $|\mathbb{F}^*_{2^{1025}}|$). This is the smallest number which divides the order of the multiplicative group corresponding to the finite field and also greater than 647. We add 1154 dummy constraints of the form $0.X_i = 0$ to make the total number of constraint 1801. Note that although the complexity of the prover algorithm depends on the number of constraints (or number of multiplications) for a specific algorithm the number of constraints may not be feasible choice for the FFT algorithm. In such case the complexity actually depends on the best possible choice of the multiplicative subgroup of $\mathbb{F}^*_{2^n}$.

This is not only applicable to MiMC or MiMCHash but a feature of the SNARK algorithm. In [BSCG+13] a finite field $\mathbb{F}_p$ is chosen in such way that $p - 1$ is of the form $2^t \cdot q$.

Over $\mathbb{F}_p$ When we use MiMC-p/p over $\mathbb{F}_p$ for some prime p (with 1025 or more bits) to construct the hash function we have the option of choosing p such that $p - 1 = 2^l \cdot q$. However this yields a very large prime number p. For ≈ 1025 bit security of the keyed permutation it is enough to have $1025/\log(3) \approx 646$ rounds. Hence the number of witness will be 1293 in this case for processing a single block. Instead of choosing such large prime we can choose p such that $p-1$ has a prime factor closed to and greater than 1293.

B More experiments with SNARK implementations of MiMC

In order to find out how the performance of the implementation changes with different-sized multipliers we performed a number of experiments which we detail here. The result is that the runtime grows very slowly, i.e. when moving from a 100-bit multiplication to a 800-bit multiplication the runtime less than doubles.

We experimented with different block sizes of the MiMC permutation in the SNARK setting, and always adjusted the required number of rounds accordingly. The time taken by the SNARK implementation for processing one block of input for different block sizes (and hence round numbers) is shown in Figure 2.

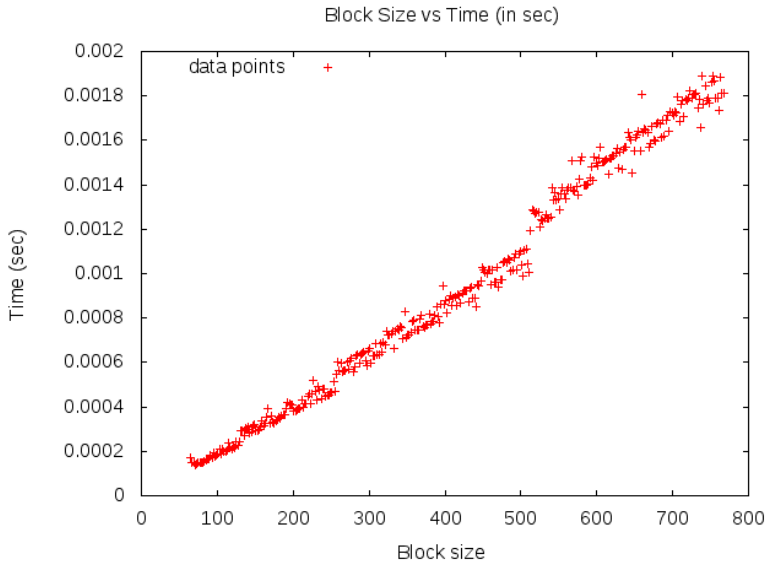

Fig. 2: Time vs. Blocksize for a SNARK implementation of MiMC

The increment of time with the block size (and hence round numbers) for SNARK implementation of MiMC appears to be almost linear. We also experimented by fixing the number of rounds of MiMC for different block sizes to a constant value, and observed the time taken to process a single block by the SNARK implementation. The result of this experiment is given in Figure 3 and shows that there is only a very slow increase in the runtime with increased block size (and hence size of the multiplier). The implementations in C++ can be found on `https://github.com/byt3bit/mimc_snark.git`.

Remark 2. For LowMC, in the constraint generation routine we only store the proper rank-1 constraints which are produced from the multiplications, as de-

scribed in definition 2. Note that the additions from the linear layer can be viewed as special rank-1 constraints where $B_i = (1, 0, \ldots, 0)$ for the i th constraint. We do not store these constraints. However, during the witness generation process these additions contribute to the complexity of the SNARK algorithm. The rank-1 constraints are necessary for the prover algorithm (see appendix E in [BSCG$^+$13]) in SNARK. Saving the constraints corresponding to the linear layers of the LowMC will only increase the constraint generation time in Table 3. We point out that this issue does not arise for MiMC in the SNARK setting.

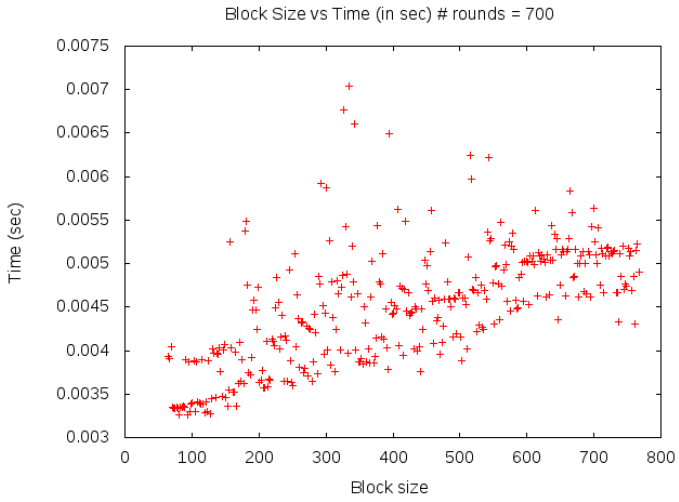

Fig. 3: Time vs. Blocksize, where the number of rounds is fixed to 700 for all block sizes