Grant Agreement Number: 825225

Safe-DEED

www.safe-deed.eu

Low complexity primitives v1

Deliverable number	*D5.2*
Dissemination level	*Public*
Delivery data	*due 29.11.2019*
Status	*Final*
Authors	*Karl Koch*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825225.

Executive Summary

The main contribution of this deliverable is the research paper *Generalized Feistel MiMC* by Albrecht et al. [AGP+19]. In this deliverable report, we give a less-technical overview of the results to make them accessible to a broader audience. For the full (technical) details, we refer to the paper.

As more and more data is stored and shared in the cloud, security and privacy of data in the cloud become more important too. One way of protecting data in the cloud is to operate only on encrypted data. Operating on encrypted data can be realized by using, for instance, secure multiparty computation (MPC). MPC gained much attention in the previous years, mainly because it is becoming practically relevant.

To make MPC practical, the choice of a suitable protocol is of paramount importance, as well as choosing a suitable underlying cipher. In this deliverable, we show and explain two ciphers suitable for MPC. Namely, *Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity* (MiMC) and *Generalized Feistel MiMC* (GMiMC). MiMC is very competitive in the area of MPC and succinct non-interactive arguments of knowledge (SNARKs). GMiMC, the successor of MiMC, further improves MiMC, leading to an even more competitive performance for MPC and SNARKs. Furthermore, GMiMC is also an interesting pseudorandom function for the post-quantum (PQ) signature scheme Picnic.

In the area of MPC, SNARKs, or PQ signature schemes like Picnic, low multiplicative complexity plays an important role. This property is provided, for example, by the ciphers MiMC and GMiMC. The design of MiMC and GMiMC led to the question if we can even further improve the performance in areas with the need for low multiplicative complexity?

Table of Contents

List of Figures

1 Introduction

Recap: Deliverable 5.1. Secure multiparty computation (MPC) becomes more and more prevalent. As described in D5.1, to make MPC practically feasible, it is of paramount importance to choose a suitable MPC protocol for the computation(s). Though, when one chooses the right MPC protocol, the computation could still be too slow for being used in practice. One reason could be the underlying cipher. Thus, to make MPC practical, it is also vital to choose a suitable cipher.

Contribution. The main contribution of this deliverable is the research paper *Generalized Feistel MiMC* by Albrecht et al. [AGP+19]. In this deliverable report, we give a less-technical overview of the results to make them accessible to a broader audience. For the full (technical) details, we refer to the paper.

In this deliverable, we present and discuss two ciphers which are suitable for MPC: (1) Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity (MiMC) [AGR+16] and (2) Generalized Feistel MiMC (GMiMC) [AGP+19]. MiMC was published in 2016 and was the basis for GMiMC. GMiMC was published in 2019 and partially created in the context of Safe-DEED.

Target Audience. The primary target of this deliverable are software engineers and researchers working with large data sets or MPC/PQ Signatures/succinct non-interactive arguments of knowledge (SNARKs). As such, we require some computer-science background for the general discussion of the topics. For an excellent introduction on some of the more technical details, we refer to the book of Smart [Sma16], and Katz and Lindell [KL14].

Outlook. In the next section, Section 2, we present the cipher Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity (MiMC). In Section 3, we present and discuss the successor of MiMC, namely Generalized Feistel MiMC (GMiMC). We highlight the main improvements of GMiMC over MiMC in Section 3.1 (MPC), Section 3.2 (Post-Quantum signatures), and Section 3.3 (SNARKs). In Section 4, we conclude this deliverable by discussing future work in the context of cipher designing for MPC. In Appendix A, we give an extended abstract of the paper by Albrecht et al. [AGP+19], which builds the basis for this deliverable report.

2 Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity (MiMC)

Few ciphers work in large prime fields (GF(p), for large p). Though in the area of secret-sharing-based MPC, this is a requirement for arithmetics on integer and fixed-point data types, respectively. This requirement was one reason to start with the creation of MiMC [AGR+16].

Design & Properties. MiMC took a relatively old design idea into consideration for its round function: $x \mapsto x^3$. This design idea was published by Nyberg and Knudsen in 1995 [NK95].

Due to this relatively simple round function, the number of non-linear operations equals the depth of the computation. This property is excellent for secret-sharing-based MPC, as in these protocols, additions are basically for free, but the performance of multiplications matters.

Applications. The class of design of MiMC is best used for cryptographic hashing or as a pseudorandom function (PRF) (block cipher). The main application of MiMC lies in SNARKs [BCG+13] (cryptographic hashing). Though, due to the performance of multiplications, MiMC is also a competitive PRF for MPC.

Performance. Grassi et al. [GRR+16] compared and benchmarked five PRFs for MPC: AES, LowMC, Naor Reingold, Legendre, and MiMC. The measurements were taken using the SPDZ framework [KSS13] with at least 1,000 PRF operations (cipher encryptions) averaged over five executions. The experiments were conducted in a LAN (1Gb/s) and WAN (50MB/s with a latency of 100ms) setting for 2 players.

Latency and *throughput* are the main metrics in the experiment. *Latency* gives the measurement of how long it takes to encrypt one block. *Throughput* shows how many blocks can be encrypted in one second. In terms of performance, besides latency, MiMC is the best PRF for MPC in this comparison. Figure 1 and Figure 2 display the results in the LAN and WAN setting, respectively.

PRF	Best latency	Best throughput		Prep. (ops/s)	Cleartext (ops/s)
	(ms/op)	Batch size	ops/s		
AES	7.713	2048	530	5.097	106268670
$F_{\mathsf{LowMC}}(\mathsf{vector})$	4.302	256	591	2.562	7000
$F_{\mathsf{LowMC}}(\mathsf{M4R})$	4.148	64	475	2.565	1420
$F_{\mathsf{NR}(128)}(\mathsf{log})$	4.375	1024	370	4.787	1359
$F_{\mathsf{NR}(128)}(\mathsf{const})$	4.549	256	281	2.384	1359
$F_{\mathsf{Leg}(\mathsf{bit})}$	0.349	2048	202969	1225	17824464
$F_{\mathsf{Leg}(1)}$	1.218	128	1535	9.574	115591
$F_{\mathsf{MiMC}}(\mathsf{basic})$	12.007	2048	8788	33.575	189525
$F_{\mathsf{MiMC}}(\mathsf{cube})$	5.889	1024	6388	33.575	189525

Figure 1: Comparison between 2-player benchmarks of PRFs for MPC: AES, LowMC, Naor Reingold, Legendre, and MiMC. The benchmarks were performed in a LAN setting. The figure is taken from the paper [GRR+16].

From the paper by Grassi et al. [GRR+16]:

> *"...one would likely prefer the Legendre PRF for applications which require low latency, and which do not involve any party external to the MPC engine, and MiMC for all other applications."*

For further details on the design of MiMC, we refer to the paper by Albrecht et al. [AGR+16]. For further details on the aforementioned performance evaluation, we refer to the paper by Grassi et al. [GRR+16].

PRF	Best latency (ms/op)	Best throughput		Prep. (ops/s)
		Batch size	ops/s	
AES	2640	1024	31.947	0.256
$F_{\mathsf{LowMC}}(\mathsf{vector})$	1315	2048	365	0.1259
$F_{\mathsf{LowMC}}(\mathsf{M4R})$	659	2048	334	0.1261
$F_{\mathsf{NR}(128)}(\mathsf{log})$	713	1024	59.703	0.2359
$F_{\mathsf{NR}(128)}(\mathsf{const})$	478	1024	30.384	0.1175
$F_{\mathsf{Leg(bit)}}$	202	1024	2053	60.241
$F_{\mathsf{Leg}(1)}$	210	512	68.413	0.4706
$F_{\mathsf{MiMC}}(\mathsf{basic})$	7379	512	59.04	1.650
$F_{\mathsf{MiMC}}(\mathsf{cube})$	3691	512	79.66	1.650

Figure 2: Comparison between 2-player benchmarks of PRFs for MPC: AES, LowMC, Naor Reingold, Legendre, and MiMC. The benchmarks were performed in a WAN setting. The figure is taken from the paper [GRR+16].

3 Generalized Feistel MiMC (GMiMC)

MiMC is a promising cipher in the domain of MPC. Though, it led to the question if MiMC can be improved? Thus, MiMC provided the basis for the creation of GMiMC.

Feistel Networks. Feistel Networks offer the excellent property that decryption can basically be applied as encryption, just with the round keys and round constants in reverse order, and the ciphertext as input. This property is especially helpful when a cipher is realized in hardware, as this approach saves space on the printed circuit board. One of the first prominent ciphers which use a Feistel Network is the Data Encryption Standard (DES) [SG16].

Typically, there are two types of Feistel Networks: (1) Balanced Feistel Networks (BFNs) and (2) Unbalanced Feistel Networks (UFNs). The two basic types differ in the fact that the branches of BFNs have the same size, $\frac{number\ of\ bits}{number\ of\ branches}$, and branches of UFNs have different sizes. Hoang et al. [HR10] give an overview of balanced, unbalanced, and other variants of Feistel networks. UFNs have been already introduced in the late 1980s. Still, UFNs have not been considered as part of a block-cipher design for a long time. UFNs were especially neglected as they were deemed to be insecure. Though, when taking the attacks on UFNs into account, they offer compelling properties for MPC-friendly ciphers, as we see in GMiMC.

Variants. GMiMC offers four different variants for the round function:

1. a contracting round function, as illustrated in Figure 3,
2. an expanding round function, as illustrated in Figure 4,
3. Nyberg's generalized Feistel network, as illustrated in Figure 5, and
4. a multi-rotating function, as illustrated in Figure 6.

All round functions are based on Feistel networks. For a detailed description of the different round functions, and even further variants within these round functions, we refer to the full paper by Albrecht et al. [AGP+19].

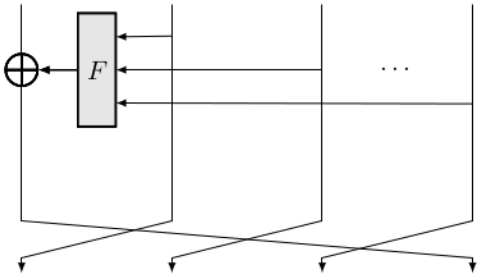

Figure 3: GMiMC's contracting round function with t branches. The figure is taken from the paper Albrecht et al. [AGP+19].

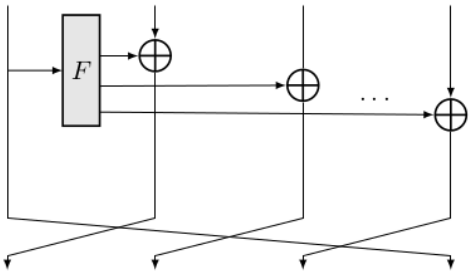

Figure 4: GMiMC's expanding round function with t branches. The figure is taken from the paper Albrecht et al. [AGP+19].

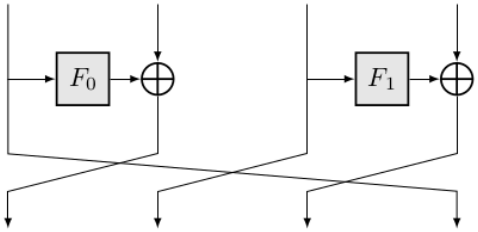

Figure 5: GMiMC's Nyberg's generalized Feistel network as round function with 4 branches. The figure is taken from the paper Albrecht et al. [AGP+19].

Security Analysis. As part of the cipher design, also a detailed security analysis has been done. As such, state-of-the-art attacks in three domains have been considered: (1) algebraic attacks, (2) statistical attacks, and (3) attacks in the post-quantum setting. For further descriptions, details, and proofs of the various attacks, we refer to the full paper by Albrecht et al. [AGP+19].

General Applications. Due to relatively low number of multiplications, ciphers like GMiMC are interesting, for instance, in the area of side-channel analysis (reduces cost of countermeasures against several side-channel attacks), homomorphic encryption (eliminates expansion of ciphertexts in homomorphic encryption schemes), SNARKs (increase throughput or latency),

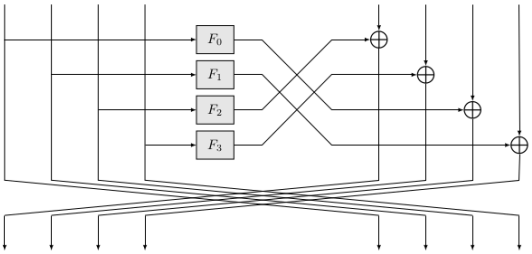

Figure 6: GMiMC's multi-rotating function as round function with 8 branches. The figure is taken from the paper Albrecht et al. [AGP+19].

or MPC (help dealing with encrypted data). Furthermore, GMiMC can also be used as a hash function, so-called *GMiMCHash*. For further details on *GMiMCHash*, we refer to the paper by Albrecht et al. [AGP+19].

Improvements over MiMC. In comparison to its predecessor MiMC, GMiMC mainly improves in three domains:

1. MPC, as further described in Section 3.1,
2. Post-Quantum (PQ) signatures, as further described in Section 3.2, and
3. SNARKs, as further described in Section 3.3.

3.1 Improvements & Applications in MPC

Framework. In terms of real-life applications using MPC, the goal was to improve the overall runtime of MPC protocols. As a framework for running MPC protocols, we chose the SPDZ framework [KSS13]. SPDZ consists of two phases: (1) a preprocessing phase and (2) an online phase. During the preprocessing phase, random so-called *Beaver* triples are generated. The procedure of generating random triples is input-independent, and thus can be performed in advance. During the online phase, these triples are consumed whenever a multiplication between shared values is performed. As the addition of shared values is almost for free in the SPDZ framework, the main bottleneck of SPDZ is the multiplication (triple generation). Hence, the aim was to improve the total number of (parallel) multiplications and the total number of operations.

Setting. We performed the experiments with two parties. For creating the benchmarks, we used the recent so-called *LowGear* protocol by Keller et al. [KPR18], which is the fastest (known) protocol for generating triples at the time of writing the paper by Albrecht et al. [AGP+19]. In terms of security of the *LowGear* protocol, we have chosen 64-bit statistical security and 128-bit computational security. In the experiment, both keys and messages were secretly shared. Each experiment comprised at least 1,000 PRF operations (block cipher calls), and we computed the average over five executions. Each block cipher was instantiated with either 4 or 16 input blocks per (Feistel) branch.

Setup. Each party had the computing power of an *Intel i7-4790 CPU* at *3.60 GHz* with *16GB of RAM*. The parties were running the protocols in a LAN network with an average round-trip time of *0.3ms* and a bandwidth of about *1GB/s*.

Measurements. To have a complete measurement of an MPC protocol, we measured both the preprocessing and the online phase. For the preprocessing phase, the amount of shared multiplications (triple generation) determines the spent time. Thus, we measure the time of creating the needed triples for one block-cipher evaluation. For the online phase, the multiplicative depth of the circuit and how often a party reveals a secret value determines the runtime. We measured the *latency* and *throughput* of the online phase. *Latency* gives the measurement of how long it takes to encrypt one block. *Throughput* shows how many blocks can be encrypted in one second.

Results. In comparison to MiMC, we could improve the throughput by a factor of more than four and reduce the preprocessing effort by a factor of five; this comes, on the other hand, at the cost of higher latency. For MPC, GMiMC's variants ERF and CRF were most competitive. We present the outcome of the experiments with the four GMiMC variants in comparison with MiMC in Figure 7. For further details on the performance evaluation, we refer to the paper by Albrecht et al. [AGP+19].

Mode		Online cost				Prep/block(ms)
	#Branch / #Block	(MPC) 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
$\text{GMiMC}_{\text{Nyb}}$		169	507	3.44	9131	5.63
$\text{GMiMC}_{\text{mrf}}$		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
$\text{GMiMC}_{\text{Nyb}}$		181	543	0.88	8605	6.03
$\text{GMiMC}_{\text{mrf}}$		183	549	1.02	8440	6.1
MiMC	16 blocks	73	3504	0.47	10780	4.86

Figure 7: MPC benchmarks of the four GMiMC variants in comparison with MiMC. The experiments were run on a commodity hardware between two players in a LAN network. Each experiment comprised at least 1,000 block cipher calls, and we computed the average over five executions. The figure is taken from the paper Albrecht et al. [AGP+19].

3.2 Improvements & Applications for PQ signatures

Picnic [CDG+17] is a PQ signature scheme based on symmetric-key primitives. Picnic's base is a one-way function f, which is used to publish an image y, the public key, of the private key x: $y = f(x)$. One of the advantages of Picnic is that it can be parameterized, and the key pairs are relatively small. When using MiMC for Picnic, the size of the signature is relatively large. Therefore, MiMC is not competitive in the area of PQ signatures. However, with the improvements in GMiMC, the size of the signature got 30 times smaller. Thus, GMiMC is an attractive option for PQ signatures, especially for Picnic.

Setting & Results. For the performance evaluation, GMiMC$_{erf}$ was used to build the one-way function f for Picnic, with a key and block size of ∼256 bits. As runtime metric, only the runtime without protocol overhead was considered. Protocol overhead is, for instance, the sampling of pseudorandom numbers. From [AGP+19]:

> *"Measuring only the circuit runtimes allows us to obtain a more accurate comparison in terms of computation time and view size,...".*

We performed the benchmarks on an *Intel Core i7-4790* at *3.6 GHz*. GMiMC$_{erf}$ outperformed MiMC in terms of runtime as well as signature size, leading to GMiMC$_{erf}$ signature sizes which are even 30 times smaller than those with MiMC. In some cases, GMiMC$_{erf}$ even performs better than the PRF LowMC. In Figure 8, we display the results in comparison with MiMC and LowMC. For further details, we refer to the paper by Albrecht et al. [AGP+19].

3.3 Improvements & Applications in SNARKs

In the area of SNARKs [BCG+13], GMiMC slightly improves over MiMC. This improvement is achieved due to the flexibility of using different field sizes in GMiMC. Furthermore, the expanding round function (ERF) seemed to be the best choice for this application.

Settings & Results. We performed the benchmarks on an *Intel Core-i7 4790* at *3.6 GHz* with a memory of *16 GB*. Each experiment got averaged over about 2,000 runs. MiMC is approximately two times better than the best competitor (at the time of writing the paper). GMiMC, in turn, is again 1.2 times better than MiMC. Figure 9 displays the improvement over MiMC.

For further details we refer to the paper by Albrecht et al. [AGP+19].

4 Conclusion

As shown in Deliverable 5.1, to make computations of secure-multiparty-computation (MPC) applications efficient, it is of paramount importance to choose a suitable protocol. In addition to a suitable protocol, it is also essential to select a suitable cipher. We have shown and described two ciphers which are suitable for MPC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity (MiMC) (by Albrecht et al. [AGR+16]) and its successor Generalized Feistel MiMC (GMiMC) (by Albrecht et al. [AGP+19]).

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
$\text{GMiMC}_{\text{erf}}$ 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
	$(16, 16, 62)$	7.59 ms	5.13 ms	1984 bits
	$(32, 8, 55)$	4.95 ms	3.05 ms	3520 bits
	$(64, 4, 81)$	11.78 ms	6.85 ms	10368 bits
	$(136, 2, 163)$	67.51 ms	35.21 ms	44336 bits
$\text{GMiMC}_{\text{erf}}$ 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

Figure 8: Comparison of GMiMC_{erf}, MiMC, and LowMC for Picnic. The figure is taken from the paper Albrecht et al. [AGP+19].

	MiMC [4]		$\text{GMiMC}_{\text{erf}}$		
$(t, \log_2(p), R)$	$(1, 1024, 646)$	$(2, 513, 647)$	$(4, 256, 332)$	$(8, 128, 178)$	$(16, 64, 141)$
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

Figure 9: Comparison of GMiMC_{erf} and MiMC for SNARKs with a block size of 1024 bits. The figure is taken from the paper Albrecht et al. [AGP+19].

MiMC & GMiMC. In addition to the area of MPC, MiMC is also a competitive cipher in the area of SNARKs. GMiMC builds upon MiMC and Feistel Networks. In the areas where MiMC is a competitive cipher, MPC and SNARKs, GMiMC can even further improve. In the area of post-quantum signatures, where MiMC is not competitive due to the relatively large signature size, GMiMC is, on the other hand, a competitive cipher.

Potential Future Work. In the area of MPC, we have conducted experiments for two players in a LAN setting. Now it would be interesting to also take, for example, more players and a WAN setting into account. Furthermore, as GMiMC is one of the first recent ciphers which considers Unbalanced Feistel Networks as well, it would be interesting to try also other variants or combinations of Feistel networks in a cipher; for instance, considering the Feistel networks shown by Hoang et al. [HR10].

References

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for mpc, and more. In *ESORICS (2)*, volume 11736 of *Lecture Notes in Computer Science*, pages 151–171. Springer, 2019.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, 2016.

[BCG+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 *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM Conference on Computer and Communications Security*, pages 1825–1842. ACM, 2017.

[GRR+16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. Mpc-friendly symmetric key primitives. In *ACM Conference on Computer and Communications Security*, pages 430–443. ACM, 2016.

[HR10] Viet Tung Hoang and Phillip Rogaway. On generalized feistel networks. In *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 613–630. Springer, 2010.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: making SPDZ great again. In *EUROCRYPT (3)*, volume 10822 of *Lecture Notes in Computer Science*, pages 158–189. Springer, 2018.

[KSS13] Marcel Keller, Peter Scholl, and Nigel P. Smart. An architecture for practical actively secure MPC with dishonest majority. In *ACM Conference on Computer and Communications Security*, pages 549–560. ACM, 2013.

[NK95] Kaisa Nyberg and Lars R. Knudsen. Provable security against a differential attack. *J. Cryptology*, 8(1):27–37, 1995.

[SG16] M. Sharma and R. B. Garg. Des: the oldest symmetric block key encryption algorithm. In *International Conference System Modeling Advancement in Research Trends (SMART)*, pages 53–58. IEEE, November 2016.

[Sma16] Nigel P. Smart. *Cryptography Made Simple*. Information Security and Cryptography. Springer, 2016.

A Feistel Networks for MPC, and More

Below we present an extended abstract from the *Generalized-Feistel-MiMC* (GMiMC) paper by Grassi et al. [GRR+16].

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 [34], Sepior [36], Sharemind [12], Unbound [37] 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 [9]. 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 [6], SCALE-MAMBA [5], BDOZa [11], VIFF [21] 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. [27] 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 [3] 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 [14, 27]. 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 [10] 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 [3] 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 speed-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 [18, 19] 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 [17, 25] 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 [3]. 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 Sect. 2.1) and use it to construct the hash function GMiMCHash (in Sect. 2.2).
We show the performance of GMiMC in MPC applications based on secret sharing techniques such as BDOZa [11], SPDZ [22] or VIFF [21]. Previous works [27, 35] 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 Sect. 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 Sect. 6.2) compared to MiMC.
In the case of the PQ signature scheme, LowMC [2, 4] 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 shown in Sect. 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 [33] 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 [20, 28], eliminate the ciphertext expansion in homomorphic encryption schemes [4,16,26,31,32], help dealing with encrypted data in secure multi-party computation systems [4, 27, 35], increase throughput or latency in SNARKs [3], 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 [13,18,23,24,29]. Research efforts in this area are manifold and cover questions on finding circuits for concrete mappings such as S-Boxes [15], foundational theoretical results on the complexity of PRGs, PRFs, and cryptographic hashes [7,8], and new ad-hoc designs of permutations, ciphers and hash functions tailored for various multiplication-related metrics [3, 4, 16, 31].

References

1. S. Agrawal, P. Mohassel, P. Mukherjee, and P. Rindal. DiSE: Distributed symmetric-key encryption. In Lie et al. [30], 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, 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.
4. 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.
5. 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/`.
6. N. Analytics. MP-SPDZ, 2019. `https://github.com/n1analytics/MP-SPDZ`.
7. 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.
8. B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.
9. 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`.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. J. Boyar, R. Peralta, and D. Pochuev. On the multiplicative complexity of Boolean functions over the basis (cap, +, 1). *Theor. Comput. Sci.*, 2000.
16. 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.
17. A. Chailloux. Quantum security of the fiat-shamir transform of commit and open protocols. *IACR Cryptology ePrint Archive*, 2019:699, 2019.
18. 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.
19. 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`.

20. J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen. Nessie Proposal: NOEKEON, 2000. http://gro.noekeon.org/Noekeon-spec.pdf.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. 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.
28. 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.
29. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In Lie et al. [30], pages 525–537.
30. D. Lie, M. Mannan, M. Backes, and X. Wang, editors. *ACM CCS 18*. ACM Press, Oct. 2018.
31. 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.
32. 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.
33. I. S. M. S. Overview. https://gist.github.com/maxvt/bb49a6c7243163b8120625fc8ae3f3cd.
34. Partisia. https://partisia.com/.
35. 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.
36. Sepior. https://sepior.com/.
37. Unbound. https://www.unboundtech.com/.