MiMC Hash

Veljko Vranić *
Final project for Zero-Knowledge Proofs course
Mathematical Academy
Belgrade, Serbia

15/12/2023

Abstract

This work aims to describe the MiMC hash function clearly and understandably.

1 Introduction

The term *hash function* is one of those terms that is omnipresent in computer science, software engineering, and related fields, that is used all over the place, but for some reason finding a person who understands them intimately is relatively hard. In this article, we try to contribute to the solution of that problem by focusing on a specific type of *cryptographic* hash functions, that are particularly important in various programmable cryptography applications (ZKPs, FHE, MPC [1]).

A brief warning note: All of the definitions and statements presented in the article are not by any means strict or technical but rather serve as an introduction to the topic.

2 Cryptographic hash functions

What are cryptographic hash functions? How are they different than *regular* hash functions?

Definition 1. *A hash function is a function that takes inputs from an arbitrary-sized set and maps them to a small fixed-sized range. (Mapping huge set to a small, fixed set)*

Example 1. *Function* $f(x) = x \mod M$*, where* $x \in \mathbb{Z}, M \in \mathbb{N}$*, lets say that* $M = 11$*. In this case, our function's range (codomain) is* $\{0 \dots 10\}$*.*

x	$f(x)$
6	6
19	8
121	0
24874	3
436292	10

As we can see in the example above, it is possible to map any number from $\mathbb{Z}$ in this way and get a number from the 'small set' (in this case $\mathbb{Z}_{11}$).

Problem 1 (Pre-image of hash)**.** *Given a hash function $f\colon A \mapsto B$ and $y \in B$, find value $x \in A$, such that $f(x) = y$.*

It is pretty obvious that given a number y from $\mathbb{Z}_{11}$, it is easy to find a number x from $\mathbb{Z}$, such that $f(x) = y$.
For instance, if we want to find such x for a number 5, number 16 is one solution for our problem.

Problem 2 (Collision problem)**.** *Given a hash function $f\colon A \mapsto B$, find values $x_1, x_2 \in A$, such that $x_1 \neq x_2$ and $f(x_1) = f(x_2)$.*

We can see that given $x \in A$ any pair of two distinct values from $\{x, x + 11, x + 2 * 11, \dots\}$ are a solution for the collision problem.

In the real world, however, we are usually trying to have the set B represent something that we can easily work with, for example, $B = \mathbb{Z}_{2^n}$ where $n \in \mathbb{N}$. This means that any number from the set B can be represented using n-bits.

Definition 2. *A cryptographic hash function is a hash function for which preimage and collision problems are computationally hard to solve.*

Computationally hard in this context can be technically defined using the n-bit security of a cryptographic primitive (encryption, hash, or some similar function).

Definition 3. *A general attack on a cryptographic system is defined as a way of computing the secret value (key) from a single pair of type (plaintext, encrypted text).*

Definition 4. *A cryptographic system offers security level n (n-bit security) if a successful general attack can be expected to require effort approximately 2^n operations [3]*

*byblos94@gmail.com

[1]ZKP - zero-knowledge proofs, FHE - fully homomorphic encryption, MCP - multi-party computation

3 Why MiMC

The efficiency of the circuits in ZK protocols depends on their algebraic structure. Comparisons between well-established cryptographic hash functions and these newer contestants (ZK-friendly hash functions: MiMC, Poseidon, Rescue) are performed as a way to avoid bottlenecks.

For example, ZK-friendly hash functions are about 50-100 times more efficient than SHA-2 when the hash is computed in zk-STARK. This huge performance gap demonstrates the need for a ZK-friendly hash function.

MiMC hashing function [1] is the first one designed specifically as a ZK-friendly cryptographic hash function. As such, it is being used frequently and is considered mature (compared to other ZK-friendly hash functions).

4 Internals

The name MiMC comes from the acronym **Mi**nimal **M**ultiplicative **C**omplexity. The inner workings of MiMC are not complex and the goal of this paper is to describe it, in detail, using a toy version of the algorithm.

All subsequent concepts will be based on the finite fields, unless stated otherwise.
The notation we will be using is $\mathbb{F}_p$, where p is a prime number. This does not cover all the finite fields (as they can also have an order of form p^n, with $n > 1$), but for the sake of simplicity, we will focus on a simpler subset of them.

4.1 Permutation polynomials

By $\mathbb{F}_p[x]$ we denote a set of all polynomials whose coefficients are elements of $\mathbb{F}_p$.

Example 2. *Let's take* $p = 11$. *In that case* $\mathbb{F}_p = \{0, \dots, 10\}$. *Some of the elements of* $\mathbb{F}_{11}[x]$ *are* $\{1, 5 + 9x, 10x^2, x^3, 4x^8 + 3x^5 + x, \dots\}$

Definition 5. *cited from [4]*
A polynomial $f \in \mathbb{F}_p[x]$ *is called a permutation polynomial of* $\mathbb{F}_p$ *if and only if one of the conditions holds:*

1. *the function* $f : \mathbb{F}_p \mapsto \mathbb{F}_p$ *is onto;*
2. *the function* $f : \mathbb{F}_p \mapsto \mathbb{F}_p$ *is one-to-one;*
3. $f(x) = a$ *has a unique solution in* $\mathbb{F}_p$ *for each* $a \in \mathbb{F}_p$*;*

Let's consider a few prime numbers and see if we can find some of the *permutation polynomials* for their finite fields.

Example 3. *For $p = 5$ and $p = 7$, let's examine a few polynomials, for example $4 + 2x$, $3x^2$ and x^3*

x	$4 + 2x$	$3x^2$	x^3
0	*4*	*0*	*0*
1	*1*	*3*	*1*
2	*3*	*2*	*3*
3	*0*	*2*	*2*
4	*2*	*3*	*4*

x	$4 + 2x$	$3x^2$	x^3
0	*4*	*0*	*0*
1	*6*	*3*	*1*
2	*1*	*5*	*1*
3	*3*	*6*	*6*
4	*5*	*6*	*1*
5	*0*	*5*	*6*
6	*2*	*3*	*6*

For the case of $\mathbb{F}_5$, $4 + 2x$ and x^3 are permutation polynomials, while $3x^2$ doesn't create a permutation of $\mathbb{F}_5$.
In the case of $\mathbb{F}_7$ we see that $4+2x$ is the permutation polynomial, while the other two are not.

This should give us an idea that in the general case, testing if the polynomial over a finite field is a permutation polynomial is not a trivial problem.

However, some of the results are well known, such as a theorem about monomials (polynomials with exactly one non-zero coefficient) and when they are permutation polynomials.

Theorem 1. *Any monomial x^d is a permutation in the field $\mathbb{F}_p$ iff $gcd(d, p - 1) = 1$.*

In order to get a better understanding of the theorem, let's try it out for monomial x^3, for which $d = 3$.
For $p = 5$, we have that $gcd(3, 4) = 1$, while for $p = 7$, we have $gcd(3, 6) = 3$.
This verifies that x^3 is a permutation polynomial for $\mathbb{F}_5$, but isn't for $\mathbb{F}_7$.

One of the corollaries of theorem 1 is that for the case of $d = 3$, our prime number p must have the form $p = 2 + 3k$, where $k \in \mathbb{N}$.

Theorem 2. *A monomial x^s is a permutation in the field $\mathbb{F}_p$ such that it inverts the effect of x^3. Such s exists and is unique because $gcd(3, p - 1) = 1$.*

5 Round function

Definition 6. *Let $F_i(x)$ be a function, defined on $\mathbb{F}_p$ such that $F_i(x) = (x+c_i)^3$, for i positive integer and c_i some random element from field $\mathbb{F}_p$. In that case, we say that such a function $F_i(x)$ is a* ***round function****.*

Values of c_i, called round constants, are chosen randomly at first and hardcoded into the implementation, so they are unchangeable.
Important note: This implies that different $MiMC$ implementations may produce different outputs!

Definition 7. *For $\mathbb{F}_p$ which has permutation polynomial x^3, let $r = \left\lceil \frac{log_2 p}{log_2 3} \right\rceil$. We call this value* ***number of rounds****.*

The number of rounds is used for getting a 'complex' permutation function, for which finding inverse permutation is computationally hard.
Finally, a composition of all these round functions produces the wanted 'complex' permutation.

Definition 8. *We call $F(x) = (F_{r-1} \circ F_{r-2} \circ \dots F_0)(x)$ 'complex' permutation function over field $\mathbb{F}_p$, where r is the number of rounds and $F_i(x)$ are round functions.*

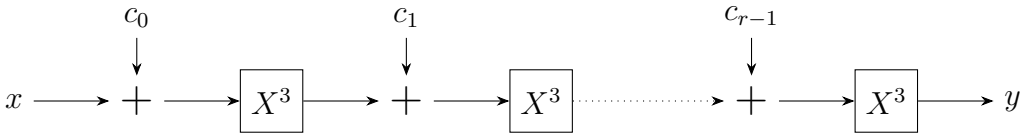

Figure 1: r rounds of MiMC

One important note about Figure 1 and the function $F(x)$ is that it has a bit different behavior in the case of $\mathbb{F}_{2^n}$.
For such fields, addition is replaced with binary operation - XOR (exclusive or). However, the topic of finite fields whose order is 2^n, even though may seem simpler, is pretty complicated, because $\mathbb{F}_{2^n}$ is not isomorphic to $\mathbb{Z}_{2^n}$, and as such has a way more complex representation.

Now, we have a permutation function whose inverse is algebraically complex. This newly generated function $F(x)$ will be used as a basis in *sponge construction*, which given a permutation creates a provably secure hash function. This method is used in many other hash functions, including SHA-3.

We might use the term *MiMC permutation* for the function $F(x)$ in subsequent text.

6 Sponge construction

Sponge construction is a family of algorithms whose input is a bitstream (array of bits) and as output gives a bitstream of the given length.

The sponge construction consists of two phases called *absorb* and *squeeze*. In the absorbing phase, the input data is absorbed into the sponge. Afterwards, the result is squeezed out.

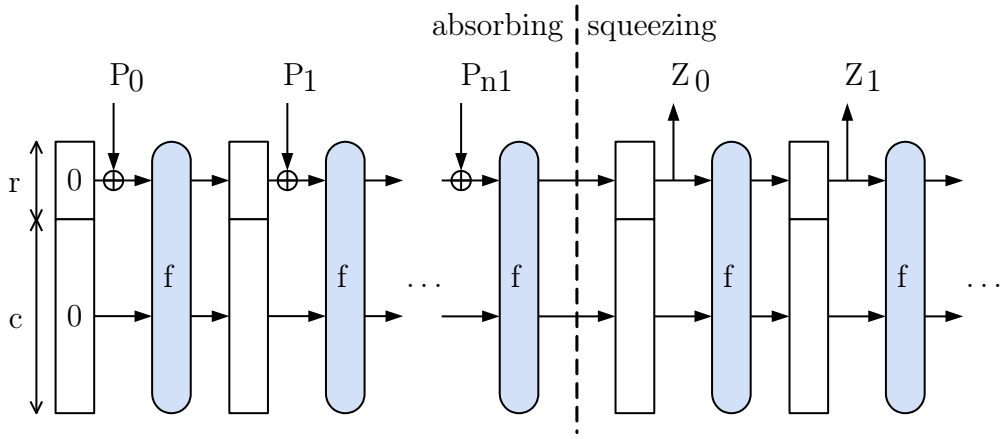

Figure 2: Sponge construction

In the absorbing phase, message blocks are added to a subset of the state, and then transformed as a whole using a permutation function (MiMC permutation in our case).
The sponge construction operates on a state of $b = r + c$ bits. The value r is called the bitrate and the value c the capacity.

The algorithm by which the sponge construction operates can be summed up in the following steps:

- Input is cut into blocks of r bits.
- b bits of the state are initialized to zero
- Absorbing phase:
  - the r-bit input blocks are XORed into the first r bits of the state, interleaved with applications of the function f
  - repeat the previous step until the end of input is reached
- Squeezing phase:
  - the first r bits of the state are returned as output blocks, interleaved with applications of the function f
  - repeat the previous step chosen number of times

7 Possible attacks

The algebraic design principle of MiMC is a reason for concern about the security of such a system. Several attack options are described in the original paper [1].

The majority of ZK hash functions typically rely on block ciphers as their foundation. When the key for a block cipher is set, it effectively becomes a permutation. It has been

mathematically established that if the underlying permutation is secure, then the hash function created from the sponge construction will also be secure.

Consequently, rather than proving the security of the hash function itself, designers focus on demonstrating the security of the block cipher that is utilized within the sponge framework.

Definition 9. *A **block cipher** is a type of symmetric-key algorithm that encrypts data in fixed-size blocks, typically of 64 or 128 bits. It operates under a set of defined rules known as the encryption key. In this process, a plaintext block is transformed into an encrypted block of ciphertext.*

We will focus only on one of the possible attacks, for the sake of the succinctness of this paper.

7.1 Interpolation attack

Definition 10. *Let $E_k \colon \mathbb{F}_p \mapsto \mathbb{F}_p$ be an encryption function. Its construction is almost the same as the construction of the MiMC permutation function, with the distinction that every round function has an additional operation with the fixed key k, as well as an additional step with k and without round constant.*

An interpolation attack constructs a polynomial corresponding to the E_k without knowledge of the secret key. If an adversary can construct such a polynomial, then for any given plaintext, the corresponding ciphertext can be produced without knowing the secret key.

Example 4. *Let's consider the case of $E_k(x)$, which has two rounds of MiMC, such that $E_k(x) = ((x+k+c_0)^3 + k + c_1)^3 + k$. This is a polynomial of degree 9 in terms of variable x.*

Once the attacker successfully creates the polynomial $E_k(x)$, they can then generate the ciphertext for any given plaintext.
This capability is essentially equivalent to possessing the encryption key itself. Furthermore, the attacker might be able to derive the key k from $E_k(x)$, as the coefficients of x^8 in the polynomial are directly related to k.

When dealing with a polynomial of degree d, the effort required to construct a Lagrange interpolation polynomial is of the order $O(d \log d)$. Consequently, it's important to use a polynomial with a sufficiently high degree to represent the encryption function.

For example, in the MiMC encryption scheme, the number of rounds r is set to be the $\lceil log_3 p \rceil$, where p is the order of the finite field $\mathbb{F}_p$. This is done to ensure that the degree d reaches its maximum possible value of $p-1$, because $d = 3^r$.

8 Circom implementation

The reference implementation of MiMC encryption/decryption algorithms, along with the hash function can be found here [2].

However, the implementation there doesn't follow the recommended 'foundational' permutation function x^3 but rather uses x^7.

The reason for such a decision was the need to perform hashing on the points on elliptic curves, most importantly on the Baby-JubJub curve.

That elliptic curve has an order $n = 8l$, where

$l = 2736030358979909402780800718157159386076813972158567259200215660948447373041$

The problem arises when we calculate $gcd(l-1, 3)$ and see that it isn't 1. As $gcd(l-1, 7) = 1$, x^7 is the permutation polynomial in over this field in the 'official' implementation.

References

[1] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *Advances in cryptology – ASIACRYPT 2016. 22nd international conference on the theory and application of cryptology and information security, Hanoi, Vietnam, December 4–8, 2016. Proceedings. Part I*, pages 191–219. Berlin: Springer, 2016.

[2] Iden3. Mimc encryption circuit, 2023.

[3] Arjen K. Lenstra. Key lengths contribution to the handbook of information security. 2010.

[4] Rudolf Lidl and Harald Niederreiter. *Finite fields.*, volume 20 of *Encycl. Math. Appl.* Cambridge: Cambridge Univ. Press, 2nd ed. edition, 1996.