MiMC in Halo2

Abstract

We give a specification of the MiMC block ciphers for implementation in the Pasta elliptic curve fields. The specification closely follows the MiMC implementation in `circom`. We also outline the proposed implementation strategy in Halo2.

1 Introduction

The MiMC block ciphers were designed to have low multiplicative complexity [1]. Two variants were proposed: one based on a cubic round function and the other having a Feistel network structure (with a cubic round function). We will refer to them as `MiMC` and `MiMC-Feistel` for convenience. The MiMC hash functions are obtained from these block ciphers by setting the key value to zero.

We are interested in MiMC implementations when the input, output, and key are from a prime field $\mathbb{F}_p$.

1.1 `MiMC` over a prime field

A `MiMC` block cipher implementation over $\mathbb{F}_p$ is specified as below:

- Set s to be smallest integer greater than one that satisfies $\gcd(s, p-1) = 1$. The gcd condition ensures that x^s is a permutation in $\mathbb{F}_p$. We want s to be the smallest such integer to keep the multiplicative complexity minimal.
- The cipher has r rounds where $r = \left\lceil \frac{\log_2 p}{\log_2 s} \right\rceil$.
- Let $\{c_i \in \mathbb{F}_p \mid i = 0, 1, \ldots, r-1\}$ be the r round constants. By convention, the first round constant is set to zero, i.e. $c_0 = 0$. The remaining round constants are derived in a pseudorandom manner and are fixed for a particular implementation.
- The ith round function is given by $F_i(x) = (x + k + c_i)^s$ where $k \in \mathbb{F}_p$ is the secret key.
- The encryption of $x \in \mathbb{F}_p$ is given by

$$\begin{aligned} E_k(x) &= F_{r-1}(\cdots F_2(F_1(F_0(x)))\cdots) + k \\ &= (\cdots((x + k + c_0)^s + k + c_1)^s \cdots + k + c_{r-1})^s + k. \end{aligned}$$

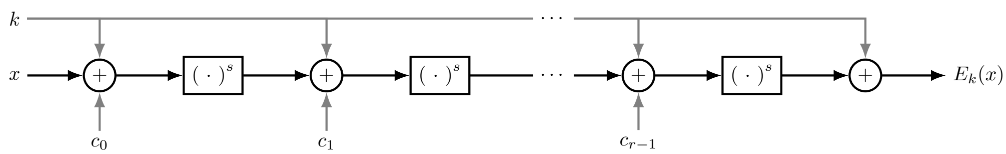

Figure 1: `MiMC` Encryption

This is illustrated in Figure 1.

1.2 `MiMC-Feistel` over a prime field

A `MiMC-Feistel` implementation over $\mathbb{F}_p$ is specified as below:

- Once again, set s to be smallest integer greater than one that satisfies $\gcd(s, p-1) = 1$.
- The cipher has r rounds where $r = 2 \times \left\lceil \frac{\log_2 p}{\log_2 s} \right\rceil$. Note that the number of rounds is twice as large as the number used in `MiMC`.
- Let $\{c_i \in \mathbb{F}_p \mid i = 0, 1, \ldots, r-1\}$ be the r round constants. By convention, the first and last round constants are set to zero, i.e. $c_0 = c_{r-1} = 0$. The remaining round constants are derived in a pseudorandom manner and are fixed for a particular implementation.
- In the first $r-1$ rounds, the ith round function is given by

  $$F_i(x_L \| x_R) = \left(x_R + (x_L + k + c_i)^s\right) \| x_L, \quad i = 0, 1, \ldots, r-1,$$

  where $x_L, x_R \in \mathbb{F}_p$ are the inputs and $k \in \mathbb{F}_p$ is the secret key.
- The last round function is given by

  $$F_{r-1}(x_L \| x_R) = x_L \| \left(x_R + (x_L + k + c_{r-1})^s\right).$$

  It does not perform the swap operation seen in previous rounds.
- The encryption of $x_L \| x_R \in \mathbb{F}_p^2$ is given by

  $$E_k(x) = F_{r-1}(\cdots F_2(F_1(F_0(x_L \| x_R)))\cdots).$$

  This is illustrated in Figure 2.

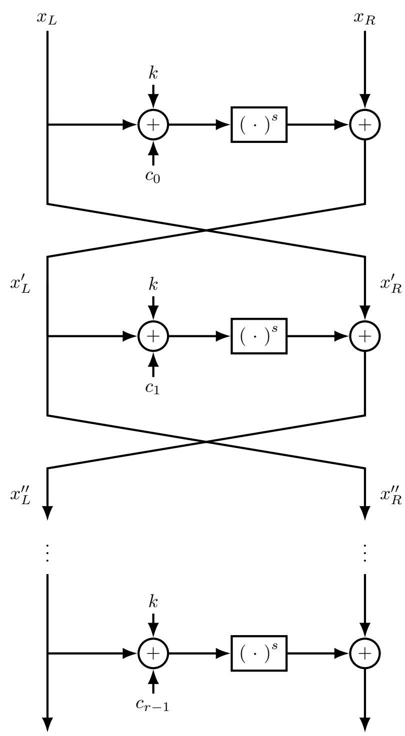

Figure 2: `MiMC` Feistel Encryption

2 MiMC in `circom`

The `circom` library [2] has implementations of both `MiMC` and `MiMC-Feistel` ciphers. The `MiMC` implementation in `circom` uses $s = 7$ while the `MiMC-Feistel` implementation uses $s = 5$.[1]

The `MiMC` implementation has 91 rounds where the round constants are elements in $\mathbb{F}_p$ where p is the order of the alt_bn128 curve. Figure 3 shows pseudocode of the round constant calculation.[2] It hashes the string "mimc"

[1]It is not clear why $s = 7$ was chosen for `MiMC` in `circom`. The field where the MiMC calculations will occur has order equal to the group order of the alt_bn128 (BN254) curve. This order is given by the prime

$$p = 21888242871839275222246405745257275088548364400416034343698204186575808495617.$$

While $\gcd(p-1,3) = 3$, we have $\gcd(p-1,5) = 1$. So $s = 5$ is a valid choice. In a document describing EdDSA on the Baby Jubjub curve [3], the authors mention that $s = 7$ is the optimal choice since $\gcd(l-1,3) = 3$ where l is group order of the Baby Jubjub curve. It turns out that $\gcd(l-1,5) = 5$ and $\gcd(l-1,7) = 1$. But the order of elliptic curve group should not matter.

[2]See `https://github.com/iden3/circomlibjs/blob/main/src/mimc7.js` for a Javascript implementation of the round constant calculation.

```
prime = 21888242871839275222246405745257275088548364400416034343698204186575808495617
num_rounds = ceil(log(prime,2)/log(7,2))

SEED = "mimc"
hash_val = keccak256(SEED)

F = GF(prime)
round_constants[0] = F(0)

for i in range(1, num_rounds):
      hash_val = keccak256(hash_val)
      round_constants[i] = F(hash_val)
```

Figure 3: `MiMC` round constant generation in `circom`

```
prime = 21888242871839275222246405745257275088548364400416034343698204186575808495617
num_rounds = 2*ceil(log(prime,2)/log(5,2))

SEED = "mimcsponge"
hash_val = keccak256(SEED)

F = GF(prime)
round_constants[0] = F(0)
round_constants[num_rounds-1] = F(0)

for i in range(1, num_rounds-1):
      hash_val = keccak256(hash_val)
      round_constants[i] = F(hash_val)
```

Figure 4: `MiMC-Feistel` round constant generation in `circom`

iteratively using the Keccak256 hash function and converts the resulting value into a field element in $\mathbb{F}_p$. By convention, the first round constant is set to zero.

The `MiMC-Feistel` implementation in `circom` has 220 rounds. The pseudocode for the round constant calculation is shown in Figure 4. The number of rounds is calculated assuming $s = 5$ and the seed string is chosen to be "mimcsponge". By convention, the first and last round constants are set to zero.

3 MiMC in Pasta Fields

Halo2 has a pair of elliptic curves called Pallas and Vesta. For

$$p = \texttt{0x40000000000000000000000000000000224698fc094cf91b992d30ed00000001},$$
$$q = \texttt{0x40000000000000000000000000000000224698fc0994a8dd8c46eb2100000001},$$

the Pallas curve is given by the equation $y^2 = x^3 + 5$ over $\mathbb{F}_p$. It forms a group of order q. The Vesta curve is given by the same equation over $\mathbb{F}_q$ and it forms a group of order p.

Since $\gcd(p-1, 3) = \gcd(q-1, 3) = 3$ and $\gcd(p-1, 5) = \gcd(q-1, 5) = 1$, we propose to use $s = 5$ in the Halo2 MiMC implementations.

The number of `MiMC` rounds in both Pasta fields will be 110, as $\left\lceil \frac{\log_2 p}{\log_2 5} \right\rceil = \left\lceil \frac{\log_2 q}{\log_2 5} \right\rceil = 110$. And the number of `MiMC-Feistel` rounds will be 220.

3.1 Round Constant Generation

We propose to use the same algorithm as `circom` to generate round constants (with the Pasta fields substituted for the alt_bn128 scalar field).

Sage codes for generating the Pallas field round constants for `MiMC` and `MiMC-Feistel` are given in Appendix A and Appendix B. These programs and the generated constants are available at `https://github.com/avras/pasta-mimc`.

4 MiMC implementation in Halo2

4.1 `MiMC` in Halo2

Columns

- One instance column to hold the input x, key k, and output $E_k(x)$ at offsets 0, 1, and 2
- One fixed column to hold the round constants; the ith row holds c_i
- One advice column where ith row holds y_i where

$$y_i = \begin{cases} x & \text{if } i = 0, \\ (y_{i-1} + k + c_{i-1})^5 & \text{if } 1 \leq i \leq 110, \\ y_{i-1} + k & \text{if } i = 111. \end{cases}$$

- Three selectors: s_1, s_2, s_3

$$\begin{aligned} s_1 = 1 &\iff i = 0 \\ s_2 = 1 &\iff 1 \leq i \leq 110 \\ s_3 = 1 &\iff i = 111 \end{aligned}$$

4.2 `MiMC-Feistel` in Halo2

Columns

- One instance column to hold the inputs x_L, x_R, key k, and output $E_k(x)$ at offsets 0, 1, 2, and 3
- One fixed column to hold the round constants; the ith row holds c_i

- Two advice columns whose ith rows hold y_i and z_i where

$$y_i \| z_i = \begin{cases} x_L \| x_R & \text{if } i = 0, \\ \left(z_{i-1} + (y_{i-1} + k + c_{i-1})^5 \right) \Big\| y_{i-1} & \text{if } 1 \le i \le 219, \\ y_{i-1} \Big\| \left(z_{i-1} + (y_{i-1} + k)^5 \right) & \text{if } i = 220. \end{cases}$$

- Three selectors: s_1, s_2, s_3

$$\begin{aligned} s_1 = 1 &\iff i = 0 \\ s_2 = 1 &\iff 1 \le i \le 219 \\ s_3 = 1 &\iff i = 220 \end{aligned}$$

A MiMC constant generation for Pallas field

```
from sage.all import *
from Crypto.Hash import keccak

prime = 0x40000000000000000000000000000000224698fc094cf91b992d30ed00000001
num_rounds = ceil(log(prime,2)/log(5,2))
print('Number of rounds =', num_rounds)

F = GF(prime)
round_constants = [F(0)]

seed_value = b'mimc';
keccak_hash = keccak.new(data=seed_value, digest_bits=256)
hash_val = keccak_hash.digest()

for i in range(1,num_rounds):
    keccak_hash = keccak.new(data=hash_val, digest_bits=256)
    hash_val = keccak_hash.digest()
    hash_val_in_hex = keccak_hash.hexdigest()

    field_element = F('0x'+hash_val_in_hex)
    round_constants.append(field_element)

print ('round_constants = [')
for i in range(num_rounds):
    print('    ', hex(round_constants[i])+',')
print(']')
```

B MiMC-Feistel constant generation for Pallas field

```
from sage.all import *
from Crypto.Hash import keccak

prime = 0x40000000000000000000000000000000224698fc094cf91b992d30ed00000001
num_rounds = 2*ceil(log(prime,2)/log(5,2))
print('Number of rounds =', num_rounds)

F = GF(prime)
round_constants = [F(0)]

seed_value = b'mimcsponge';
keccak_hash = keccak.new(data=seed_value, digest_bits=256)
hash_val = keccak_hash.digest()

for i in range(1,num_rounds-1):
    keccak_hash = keccak.new(data=hash_val, digest_bits=256)
    hash_val = keccak_hash.digest()
    hash_val_in_hex = keccak_hash.hexdigest()

    field_element = F('0x'+hash_val_in_hex)
    round_constants.append(field_element)
round_constants.append(F(0))

print ('round_constants = [')
for i in range(num_rounds):
    print('    ', hex(round_constants[i])+',')
print(']')
```

References

[1] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. Cryptology ePrint Archive, Paper 2016/492, 2016. https://eprint.iacr.org/2016/492.

[2] CircomLib. https://github.com/iden3/circomlib.

[3] Jordi Baylina and Marta Bellés. EdDSA for Baby Jubjub elliptic curve with MiMC-7 hash. https://iden3-docs.readthedocs.io/en/latest/_downloads/a04267077fb3fdbf2b608e014706e004/Ed-DSA.pdf.