Algebraic Attacks against Some Arithmetization-Oriented Hash Functions

Augustin Bariant

Inria Paris, COSMIQ team

September 20, 2022

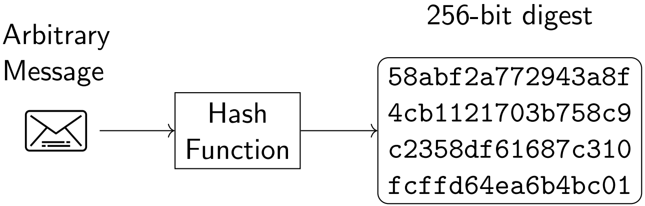
1 Introduction to symmetric cryptography

- Cryptographic Hash Functions
- The sponge construction and the CICO problem
- Arithmetization-oriented Ciphers

2 Algebraic Cryptanalysis

- Modelling the CICO Problem
- An Efficient GCD Algorithm
- Algebraic cryptanalysis of Feistel-MiMC and Poseidon

Cryptographic Hash Functions

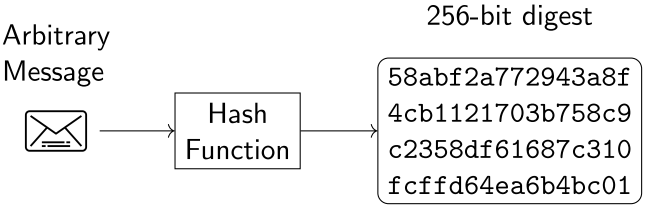


Cryptographic Hash Function

Deterministic function with the following security properties:

- Pre-image resistance: Difficult to invert.
- Second pre-image resistance: Given a message and its digest, difficult to find a second message with the same digest.
- Collision resistance: Difficult to find any two messages with the same digest.

Cryptographic Hash Functions: Insights



- Brute-force preimage attack: Hash random messages until the given digest is found. Complexity in $O(2^n)$ for a n-bit digest.
- Brute-force collision attack: Hash random messages and store their digest in a hashtable, until a collision is found. Complexity in $O(2^{n/2})$ for a n-bit digest.
- Usually, the digest size is ≥ 256.

Cryptographic Hash Functions: Applications (1)

Famous Hash Algorithms: SHA1 (broken), MD5 (broken), SHA256, SHA3. . .

Cryptographic Hash Functions: Applications (1)

Famous Hash Algorithms: SHA1 (broken), MD5 (broken), SHA256, SHA3...

- Password storage: Databases store passwords hash instead of clear passwords: in case of a database leak, it doesn't fully compromise the users credentials.

Cryptographic Hash Functions: Applications (1)

Famous Hash Algorithms: SHA1 (broken), MD5 (broken), SHA256, SHA3...

- Password storage: Databases store passwords hash instead of clear passwords: in case of a database leak, it doesn't fully compromise the users credentials.
- File integrity verification: Comparing a downloaded-file hash to a certified hash ensures that the correct file has been downloaded.
    - Ex: the TLS protocol in HTTPS verifies the data integrity and authenticity with hash functions.

Cryptographic Hash Functions: Applications (1)

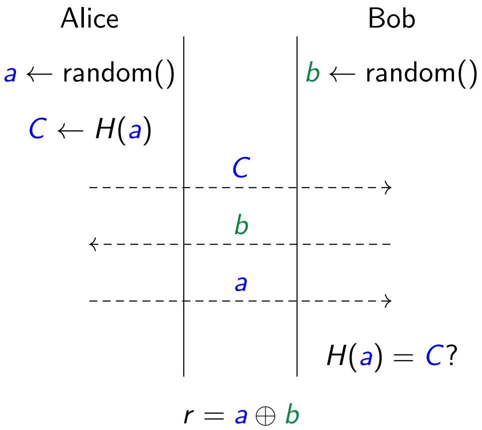
Famous Hash Algorithms: SHA1 (broken), MD5 (broken), SHA256, SHA3...

- Password storage: Databases store passwords hash instead of clear passwords: in case of a database leak, it doesn't fully compromise the users credentials.
- File integrity verification: Comparing a downloaded-file hash to a certified hash ensures that the correct file has been downloaded.
  - Ex: the TLS protocol in HTTPS verifies the data integrity and authenticity with hash functions.
- Proof of work (blockchain): Finding a message with a constrained digest (e.g. starting with k zeros) is costly (e.g. $O(2^k)$ hashs), so that a malicious user needs an excessively huge computational power to attack the blockchain.

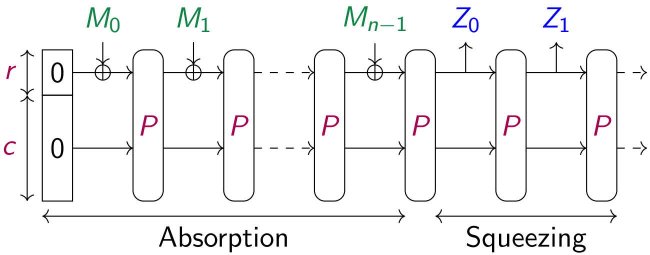
Cryptographic Hash Functions: Applications (2)

Coin Flipping protocol:

- Alice and Bob don't trust each other.
- They wish to agree on an unbiased random number.
- Alice commits a using the hash function H.
- $r = a \oplus b$ can't be biased by either party if H is a secure cryptographic hash function.



A Hash function framework: the sponge construction



The sponge construction

- **Parameters:** A public permutation P, a rate r and a capacity c.
- **Input:** A message split into n blocks M_i of r bits.
- **Output:** An arbitrarily long hash sequence Z_i.

Towards an ideal public permutation

- An ideal permutation is a permutation that looks like a random permutation.
- It is often constructed using an iterated round function:

$$P = f \circ f \circ \cdots \circ f = f^{(R)}$$

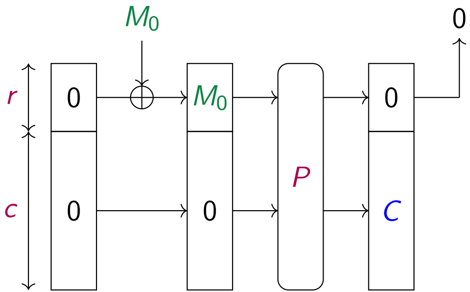
- An ideal permutation should be strong against the CICO problem:

The Constrained Input Constrained Output (CICO) Problem

Find x,y such that $P(x\|0) = (y\|0)$.

The CICO problem against the sponge construction

- Suppose that we know a r-bit M_0 and C such that $P(M_0 \| 0) = 0 \| C$.
- M_0 is a preimage of the r-bit digest $Z = 0$ (one output block):



Cryptanalysis of a public permutation

How do we know if a public permutation behaves like an ideal permutation?

Cryptanalysis of a public permutation

How do we know if a public permutation behaves like an ideal permutation?

- We don't.

Cryptanalysis of a public permutation

How do we know if a public permutation behaves like an ideal permutation?

- We don't.
- Cryptographers try to find unwanted properties, such as CICO solutions.
- Study round-reduced versions $P_i = f^{(i)}$, $i \leq R$.
- Confidence in the permutation gained with a lot of cryptanalysis.

Cryptanalysis of a public permutation

How do we know if a public permutation behaves like an ideal permutation?

- We don't.
- Cryptographers try to find unwanted properties, such as CICO solutions.
- Study round-reduced versions $P_i = f^{(i)}$, $i \leq R$.
- Confidence in the permutation gained with a lot of cryptanalysis **(or not)**.

In this presentation, we study public permutations of some non-traditional ciphers.

Traditional vs Arithmetization-oriented ciphers

Traditional ciphers

- Designed for bit-oriented platforms (computers, chips, ASIC. . .).

Arithmetization-oriented ciphers

- Designed for Zero-Knowledge Proofs and Multi Party Computation protocols.

Traditional vs Arithmetization-oriented ciphers

Traditional ciphers

- Designed for bit-oriented platforms (computers, chips, ASIC. . .).
- Operate on bit sequences.
  All operations are allowed.

Arithmetization-oriented ciphers

- Designed for Zero-Knowledge Proofs and Multi Party Computation protocols.
- Operate on large finite fields.
  $+$ and $\times$ operations are allowed.

Traditional vs Arithmetization-oriented ciphers

Traditional ciphers

- Designed for bit-oriented platforms (computers, chips, ASIC...).
- Operate on bit sequences.
  All operations are allowed.
- Designed to minimize the resource consumption (time, memory...).

Arithmetization-oriented ciphers

- Designed for Zero-Knowledge Proofs and Multi Party Computation protocols.
- Operate on large finite fields.
  $+$ and $\times$ operations are allowed.
- Designed to minimize the number of (sequential) multiplications.

Traditional vs Arithmetization-oriented ciphers

Traditional ciphers

- Designed for bit-oriented platforms (computers, chips, ASIC. . .).
- Operate on bit sequences. All operations are allowed.
- Designed to minimize the resource consumption (time, memory. . .).
- Several decades of cryptanalysis.

Arithmetization-oriented ciphers

- Designed for Zero-Knowledge Proofs and Multi Party Computation protocols.
- Operate on large finite fields. $+$ and $\times$ operations are allowed.
- Designed to minimize the number of (sequential) multiplications.
- 5 years of cryptanalysis.

Arithmetization-oriented ciphers operate on finite fields.

But what is a finite field?

Finite fields

- A field is a set $\mathbb{K}$ with:
  - A $+$ operation (commutative, associative and all elements have an inverse) and a neutral element 0.
  - A $\times$ operation (commutative, associative, and distributive for the addition) and a neutral element 1.
  - All elements of $\mathbb{K} \setminus \{0\}$ have an inverse for $\times$.
- Ex: $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{C}$, $\mathbb{Z}/p\mathbb{Z}$ with p prime...

Finite fields

- A field is a set $\mathbb{K}$ with:
  - A $+$ operation (commutative, associative and all elements have an inverse) and a neutral element 0.
  - A $\times$ operation (commutative, associative, and distributive for the addition) and a neutral element 1.
  - All elements of $\mathbb{K} \setminus \{0\}$ have an inverse for $\times$.
- Ex: $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{C}$, $\mathbb{Z}/p\mathbb{Z}$ with p prime...
- For any prime p and integer $e \geq 1$, a field of size $q = p^e$ exists (called $\mathbb{F}_q$).

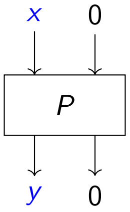
In this talk, Algorithms operate on $\mathbb{F}_p$ with $p \approx 2^{64}$ prime: $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$.

Goal: Study the CICO Problem on several permutations

We study permutations of $\mathbb{F}_p^2$ with $p = 18446744073709551557 = 2^{64} - 59$.

Constrained Input Constrained Output (CICO) Problem

Find $x, y \in \mathbb{F}_p$ such that $P(x, 0) = (y, 0)$.



ZK Hash Function Cryptanalysis Challenge

- Challenge launched by the Ethereum Fundation in November 2021.
- 4 Arithmetization-oriented hash functions under study: Feistel–MiMC, Poseidon, Rescue–Prime and Reinforced Concrete.
- Goal: crack the CICO problem on reduced versions of them.

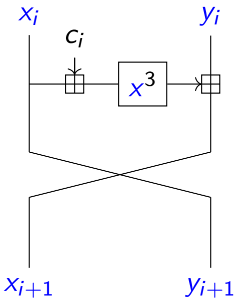
ZK Hash Function Cryptanalysis Challenge

- Challenge launched by the Ethereum Fundation in November 2021.
- 4 Arithmetization-oriented hash functions under study: Feistel–MiMC, Poseidon, Rescue–Prime and Reinforced Concrete.
- Goal: crack the CICO problem on reduced versions of them.

Total Bounty Budget: $200 000.

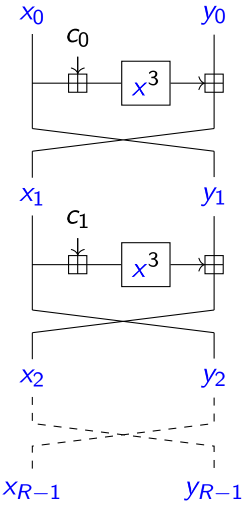
Presentation of Feistel-MiMC

$$\begin{cases} x_{i+1} = (x_i + c_i)^3 + y_i \\ y_{i+1} = x_i \end{cases}$$



- Round function iterated R times.
- $R = 80$ in the full version.
- Challenges go from 6 to 40 rounds.
- How to we solve the CICO problem?

Presentation of Feistel-MiMC



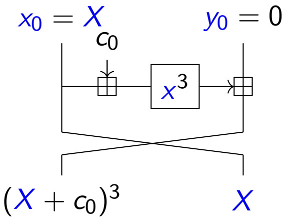
- Round function iterated R times.
- $R = 80$ in the full version.
- Challenges go from 6 to 40 rounds.
- How to we solve the CICO problem?

The CICO Problem with Feistel-MiMC

$x_0 = X$ $\quad$ $y_0 = 0$

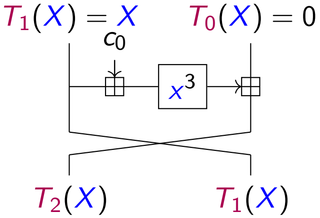
- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).

The CICO Problem with Feistel-MiMC



- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).

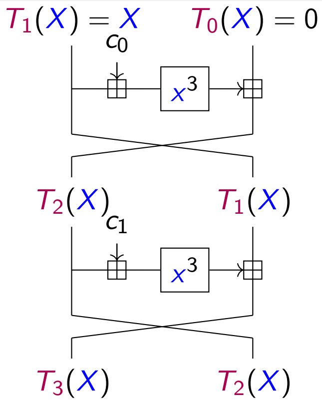
The CICO Problem with Feistel-MiMC



- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).
- Define $T_i(X)$ with the following:

$$\begin{cases} T_0(X) = y_0 = 0 \\ T_1(X) = x_0 = X \\ T_{i+1}(X) = (T_i(X) + c_{i-1})^3 + T_{i-1}(X) \end{cases}$$

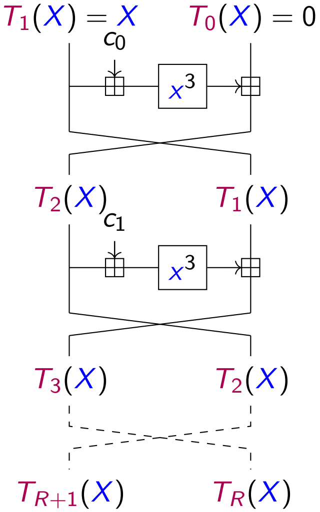
The CICO Problem with Feistel-MiMC



- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).
- Define $T_i(X)$ with the following:

$$\begin{cases} T_0(X) = y_0 = 0 \\ T_1(X) = x_0 = X \\ T_{i+1}(X) = (T_i(X) + c_{i-1})^3 + T_{i-1}(X) \end{cases}$$

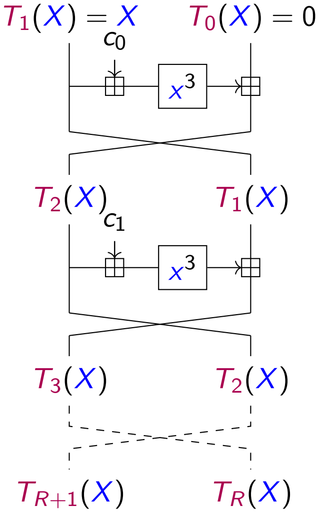
The CICO Problem with Feistel-MiMC



- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).
- Define $T_i(X)$ with the following:

$$\begin{cases} T_0(X) = y_0 = 0 \\ T_1(X) = x_0 = X \\ T_{i+1}(X) = (T_i(X) + c_{i-1})^3 + T_{i-1}(X) \end{cases}$$

The CICO Problem with Feistel-MiMC

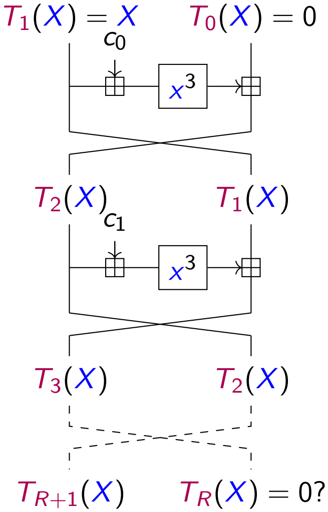


- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).
- Define $T_i(X)$ with the following:

$$\begin{cases} T_0(X) = y_0 = 0 \\ T_1(X) = x_0 = X \\ T_{i+1}(X) = (T_i(X) + c_{i-1})^3 + T_{i-1}(X) \end{cases}$$

- By induction, T_R is of degree 3^{R-1}.

The CICO Problem with Feistel-MiMC



- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).
- Define $T_i(X)$ with the following:

$$\begin{cases} T_0(X) = y_0 = 0 \\ T_1(X) = x_0 = X \\ T_{i+1}(X) = (T_i(X) + c_{i-1})^3 + T_{i-1}(X) \end{cases}$$

- By induction, T_R is of degree 3^{R-1}.
- Find a root r of T_R.
- $(r, 0) \to (T_{R+1}(r), 0)$ solves CICO.
- How much does it cost to find r?

Remarks on polynomials in $\mathbb{F}_p$

- Some polynomials have no root in $\mathbb{F}_p$ ($\mathbb{F}_p$ is not algebraically closed, like $\mathbb{R}$).
- All elements of $\mathbb{F}_p$ are roots of $X^p - X$ $(= \prod_{\omega \in \mathbb{F}_p}(X - \omega))$.
- Therefore, $T(X)$ has a root in $\mathbb{F}_p$ iff $T(X)$ and $X^p - X$ have a common factor.

Remarks on polynomials in $\mathbb{F}_p$

- Some polynomials have no root in $\mathbb{F}_p$ ($\mathbb{F}_p$ is not algebraically closed, like $\mathbb{R}$).
- All elements of $\mathbb{F}_p$ are roots of $X^p - X$ ($= \prod_{\omega \in \mathbb{F}_p}(X - \omega)$).
- Therefore, $T(X)$ has a root in $\mathbb{F}_p$ iff $T(X)$ and $X^p - X$ have a common factor.

Idea of the root-finding algorithm on $T(X)$ (of degree $d \ll p$):

- Compute the Greatest Common Divisor (GCD) of $X^p - X$ and $T(X)$.

  $\rightarrow$ The GCD is of low degree in average.
- Factorize it if needed.

A Greatest Common Divisor (GCD) algorithm

- Common divisors are given with the Euclidian GCD algorithm:
  - Given U, V two polynomials of degrees d_u, d_v such that $d_u \geq d_v$, find Q, R such that:

    $$U = Q \cdot V + R$$

    with $\deg(R) < d_v$.
  - Set $U, V = V, R$ and iterate.
  - If $R = 0$, return U.

A Greatest Common Divisor (GCD) algorithm

- Common divisors are given with the Euclidian GCD algorithm:
  - Given U, V two polynomials of degrees d_u, d_v such that $d_u \geq d_v$, find Q, R such that:
    $$U = Q \cdot V + R$$
    with $\deg(R) < d_v$.
  - Set $U, V = V, R$ and iterate.
  - If $R = 0$, return U.
- Apply the algorithm with $U(X) = X^p - X$ and $V(X) = T(X)$ (degree $d \ll p$).
- Q from the first step is of degree $p - d \approx 2^{64}$ and cannot be computed.
- Observation: We only need R of the first step.

An improved first step of the Euclidian GCD algorithm

Goal: find R such that $X^p - X = QT + R$ and $\deg(R) < \deg(T)$.

- Equivalently, $R = X^p - X \mod T$.
- We compute $X^p \mod T$ recursively using fast exponentation:

$$\begin{cases} X^k = 1 & \text{if } k = 0 \\ X^k = (X^{\frac{k}{2}})^2 \mod T & \text{if } k \text{ is even} \\ X^k = (X^{\frac{k-1}{2}})^2 \times X \mod T & \text{if } k \text{ is odd} \end{cases}$$

- $\log_2(p)$ steps to compute $X^p \mod T$. Deduce $\mathrm{R} = X^p - X \mod T$.

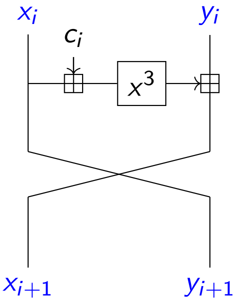
Root-finding Algorithm of a Polynomial in $\mathbb{F}_p$

Goal: Find the roots of $T(X)$ of degree $d \ll p$ in $\mathbb{F}_p$.

- Compute $R(X) = X^p - X \mod T(X)$ using fast exponentiation.
- Compute $G(X) = \gcd(T, R)$ using efficient euclidian GCD algorithm.
- Factorize $G(X)$.
- In total, it costs $O(d \log(d) \log(p))$, and is practical up to $d = 2^{32}$.

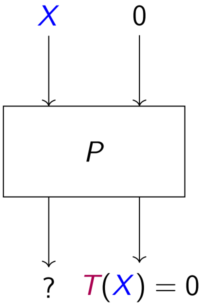
$\rightarrow$ We can break 21 rounds of Feistel-MiMC experimentally (out of 80 rounds).

Summary of the CICO cryptanalysis on Feistel-MiMC



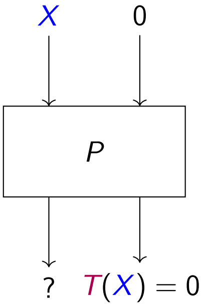
- Low degree round function (degree 3).

Summary of the CICO cryptanalysis on Feistel-MiMC



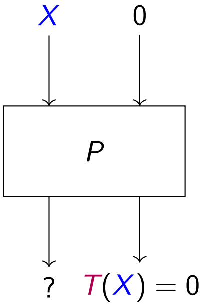
- Low degree round function (degree 3).
- Modelize CICO with a root-finding problem.

Summary of the CICO cryptanalysis on Feistel-MiMC



- Low degree round function (degree 3).
- Modelize CICO with a root-finding problem.
- The solve complexity is quasi-linear in the degree ($O(d \log(d) \log(p))$).

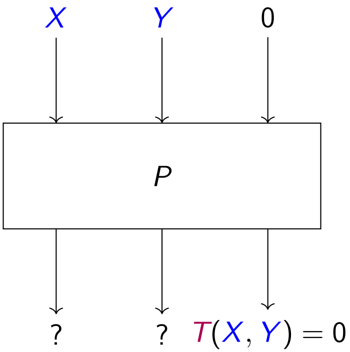
Summary of the CICO cryptanalysis on Feistel-MiMC



- Low degree round function (degree 3).
- Modelize CICO with a root-finding problem.
- The solve complexity is quasi-linear in the degree ($O(d \log(d) \log(p))$).
- The degree depends on the number of rounds: $d = 3^{R-1}$.

For a security level of 64 bits, 40 rounds are necessary.

The CICO Problem with Poseidon (over $\mathbb{F}_p^3$)



- Low degree round function.
- Set Y to a constant (e.g. 0) and solve $T(X, 0) = 0$.
- It works because T is of degree $d \ll p$.

Conclusion

- We study public permutations on big finite fields with the CICO problem.
- The CICO problem is a root-finding problem.
- We estimate the complexity of the attack with the best root-finding algorithm.
- We deduce a lower bound on the number of rounds for a given security level.
- Lead to the publication of a paper in Transactions on Symmetric Cryptography with Clémence Bouvier, Gaëtan Leurent & Léo Perrin.

Conclusion

- We study public permutations on big finite fields with the CICO problem.
- The CICO problem is a root-finding problem.
- We estimate the complexity of the attack with the best root-finding algorithm.
- We deduce a lower bound on the number of rounds for a given security level.
- Lead to the publication of a paper in Transactions on Symmetric Cryptography with Clémence Bouvier, Gaëtan Leurent & Léo Perrin.

Thank you for your attention.