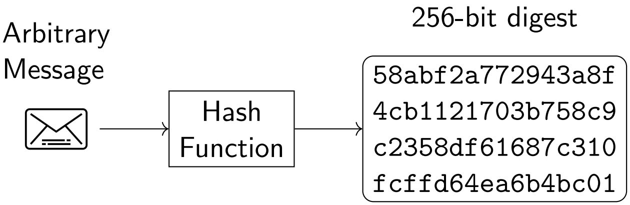
Algebraic Attacks against Some Arithmetization-Oriented Symmetric Cryptographic Algorithms

Augustin Bariant

Inria Paris, COSMIQ team

November 24, 2022

Cryptographic Hash Functions

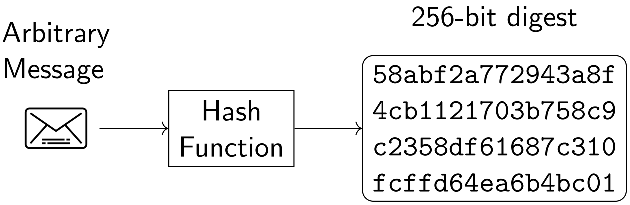


Cryptographic Hash Function

Deterministic random-looking 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: MD5 (broken), SHA1 (broken), SHA256, SHA3...

Cryptographic Hash Functions: Applications (1)

Famous Hash Algorithms: MD5 (broken), SHA1 (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: MD5 (broken), SHA1 (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.
- Signature: When signing a message (e.g. with RSA), first hash the message, then sign the hash.
  - Ex: the TLS protocol in HTTPS verifies the data integrity and authenticity with hash functions.

Cryptographic Hash Functions: Applications (1)

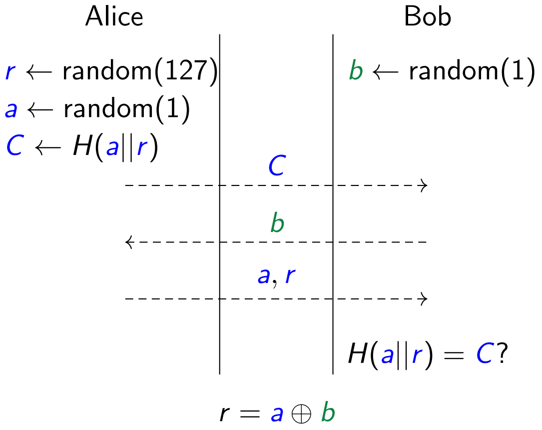
Famous Hash Algorithms: MD5 (broken), SHA1 (broken), SHA256, SHA3...

- Password storage: Databases store passwords hash instead of clear passwords: in case of a database leak, it doesn't fully compromise the users credentials.
- Signature: When signing a message (e.g. with RSA), first hash the message, then sign the hash.
    - 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.

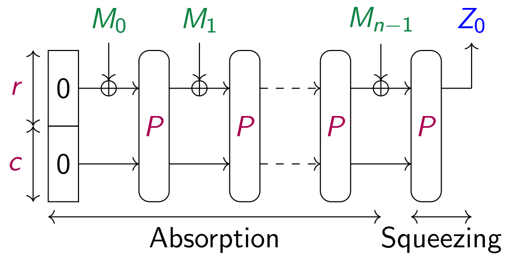
Cryptographic Hash Functions: Applications (2)

Coin Flipping protocol:

- Alice and Bob don't trust each other.
- They wish to agree on an unbiased random bit.
- Alice commits a using a large random value r and 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:** A hash block Z_0.

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 (like block ciphers):

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

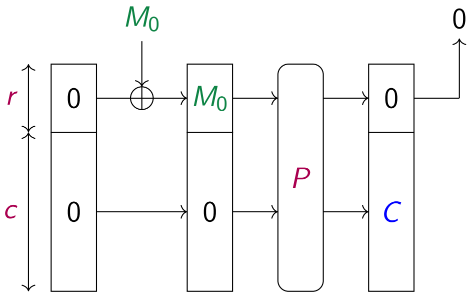
- An ideal permutation should be strong against the CICO problem:

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 public permutations

How do we study public permutations, such as public permutations?

Cryptanalysis of public permutations

How do we study public permutations, such as public permutations?

- We can't prove their security.

Cryptanalysis of public permutations

How do we study public permutations, such as public permutations?

- We can't prove their security.
- Cryptographers try to find unwanted properties, such as CICO solutions.
- Study round-reduced versions $P_i = f^{(i)}$, $i \leq R$.

Cryptanalysis of public permutations

How do we study public permutations, such as public permutations?

- We can't prove their security.
- Cryptographers try to find unwanted properties, such as CICO solutions.
- Study round-reduced versions $P_i = f^{(i)}$, $i \leq R$.
- Cryptanalysis works give an idea of the number of rounds that resists to known attacks.
- Designers then add more rounds as a security margin.
- They chose a tradeoff between security trust and performance.

Cryptanalysis of public permutations

How do we study public permutations, such as public permutations?

- We can't prove their security.
- Cryptographers try to find unwanted properties, such as CICO solutions.
- Study round-reduced versions $P_i = f^{(i)}$, $i \leq R$.
- Cryptanalysis works give an idea of the number of rounds that resists to known attacks.
- Designers then add more rounds as a security margin.
- They chose a tradeoff between security trust and performance.

In this presentation, we study public permutations, but in a non-traditional context.

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.
  Permutations of ≈ 256 bits.

Arithmetization-oriented ciphers

- Designed for Zero-Knowledge Proofs and Multi Party Computation protocols.
- Operate on large finite fields $\mathbb{F}_p$.
  $+$ and $\times$ operations are allowed.
  Permutations of $\mathbb{F}_p^m$ $(m = 2, 3 \ldots)$

Traditional vs Arithmetization-oriented ciphers

Traditional ciphers

- Designed for bit-oriented platforms (computers, chips, ASIC...).
- Operate on bit sequences.
  All operations are allowed.
  Permutations of ≈ 256 bits.
- 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 $\mathbb{F}_p$.
  $+$ and $\times$ operations are allowed.
  Permutations of $\mathbb{F}_p^m$ $(m = 2, 3 \ldots)$
- 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.
  Permutations of ≈ 256 bits.
- 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 $\mathbb{F}_p$.
  $+$ and $\times$ operations are allowed.
  Permutations of $\mathbb{F}_p^m$ $(m = 2, 3 \dots)$
- Designed to minimize the number of (sequential) multiplications.
- 5 years of cryptanalysis.

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.
- Specific parameters (p, m in $\mathbb{F}_p^m \dots$) were chosen by Ethereum.
- Goal: solve the CICO problem on reduced versions of them.
- Versions are designed to be (almost) in range of practical attacks.

ZK Hash Function Cryptanalysis Challenge

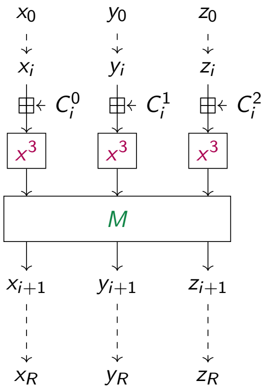
- Challenge launched by the Ethereum Fundation in November 2021.
- 4 Arithmetization-oriented hash functions under study: Feistel–MiMC, Poseidon, Rescue–Prime and Reinforced Concrete.
- Specific parameters (p, m in $\mathbb{F}_p^m \dots$) were chosen by Ethereum.
- Goal: solve the CICO problem on reduced versions of them.
- Versions are designed to be (almost) in range of practical attacks.

Total Bounty Budget: $200 000.

Poseidon: an example of Arithmetization-oriented cipher

Poseidon is a permutation of $\mathbb{F}_p^3$.

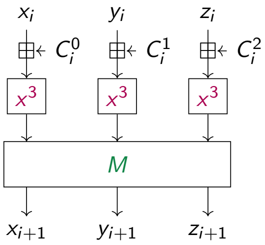
- $p \approx 2^{64}$ is the size of the field in the Ethereum challenges.
- p determines the security level.
- $x \to x^3$ is the non-linear component:
  - Invertible if $p - 1$ is divisible by 3, and of inverse $x \to x^{(p-1)/3}$ since $x^{p-1} = 1$.
- M is a 3x3 linear matrix.



Poseidon: an example of Arithmetization-oriented cipher

Poseidon is a permutation of $\mathbb{F}_p^3$.

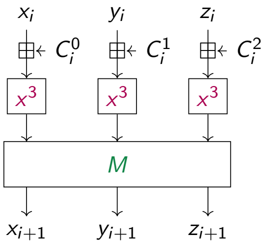
- $p \approx 2^{64}$ is the size of the field in the Ethereum challenges.
- p determines the security level.
- $x \to x^3$ is the non-linear component:
  - Invertible if $p-1$ is divisible by 3, and of inverse $x \to x^{(p-1)/3}$ since $x^{p-1} = 1$.
- M is a 3x3 linear matrix.



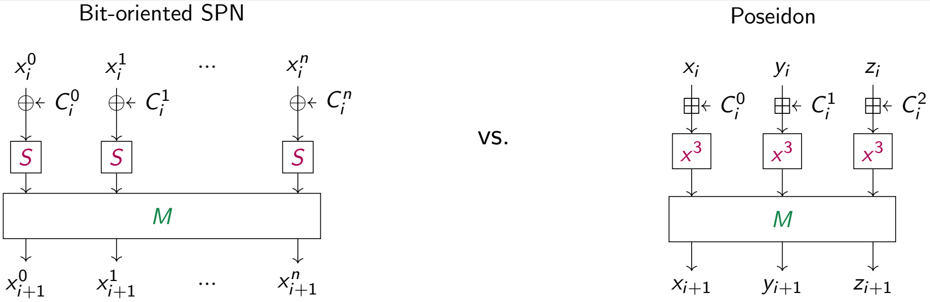
Poseidon: an example of Arithmetization-oriented cipher

Poseidon is a permutation of $\mathbb{F}_p^3$.

- $p \approx 2^{64}$ is the size of the field in the Ethereum challenges.
- p determines the security level.
- $x \to x^3$ is the non-linear component:
  - Invertible if $p-1$ is divisible by 3, and of inverse $x \to x^{(p-1)/3}$ since $x^{p-1} = 1$.
- M is a 3x3 linear matrix.
- Substitution-Permutation Network (SPN) cipher.



Poseidon: an example of Arithmetization-oriented cipher



- n nibbles of 4 to 8 bits for bit-oriented SPN vs 3 elements of $\mathbb{F}_p$ for Poseidon.
- $x \to x^3$ is a strong and big Sbox:

$$\max_{\delta_i, \delta_o \in \mathbb{F}_p} \Pr_{x \in \mathbb{F}_p} \left((x + \delta_i)^3 - x^3 = \delta_o \right) \leq \frac{2}{p} \approx 2^{-63}$$

Insight on Algebraic Attacks

- Arithmetization-based ciphers resist well to classic attacks (e.g. differential, linear. . .).
- But Poseidon Sbox $x \to x^3$ has a low degree in a field, which is often not the case in bit-oriented SPN ciphers.

Algebraic attacks exploit these low degree S-boxes.

Algebraic Attacks: How they work

- Modelize the CICO problem with a set of polynomials in $\mathbb{F}_p$.
  - A non-trivial solution $X_1, \dots X_n$ should provide sufficient information to solve the problem.

$$\begin{cases} T_1(X_1, \dots X_n) = 0 \\ T_2(X_1, \dots X_n) = 0 \\ \vdots \\ T_n(X_1, \dots X_n) = 0 \end{cases}$$

Algebraic Attacks: How they work

- Modelize the CICO problem with a set of polynomials in $\mathbb{F}_p$.
  - A non-trivial solution $X_1, \ldots X_n$ should provide sufficient information to solve the problem.
- Solve the polynomial system.
- From the values $x_1, \ldots x_n$, build a CICO solution.

$$\begin{cases} T_1(X_1, \ldots X_n) = 0 \\ T_2(X_1, \ldots X_n) = 0 \\ \vdots \\ T_n(X_1, \ldots X_n) = 0 \end{cases}$$

$$\downarrow$$

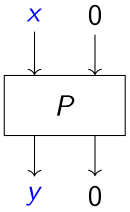
$$X_1 = x_1$$
$$\vdots$$
$$X_n = x_n$$

Example: the CICO Problem on Feistel-MiMC.

Feistel-MiMC is a permutation of $\mathbb{F}_p^2$ with p the largest prime $\leq 2^{64}$.

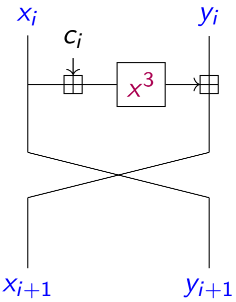
Constrained Input Constrained Output (CICO) Problem

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



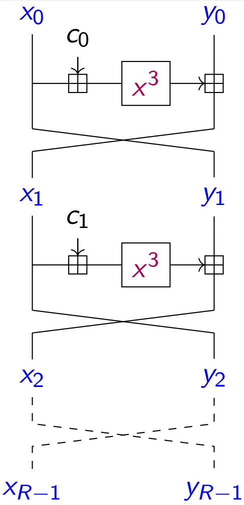
Description of Feistel-MiMC

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



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

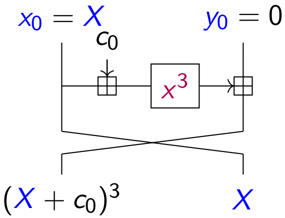
Description of Feistel-MiMC



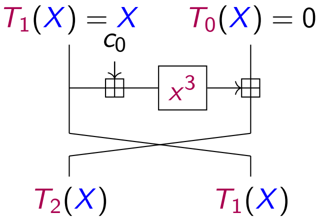
- Round function iterated R times.
- $R = 80$ in the full version.
- Ethereum challenges go from 6 to 40 rounds.
- How do we solve the CICO problem?

$x_0 = X$ $\qquad y_0 = 0$

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

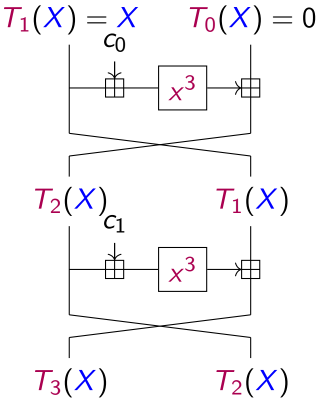


- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).



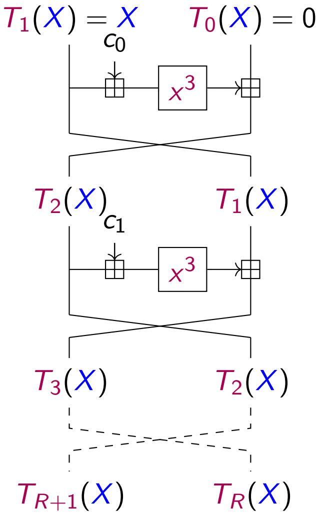
- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).
- Define $T_i(X)$ with the following:

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



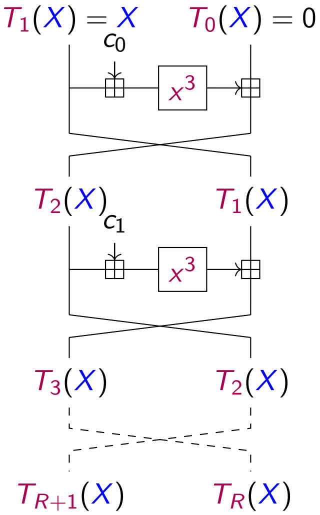
- Define a variable X representing x_0.
- Set $y_0 = 0$ (Contrained Input).
- Define $T_i(X)$ with the following:

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



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

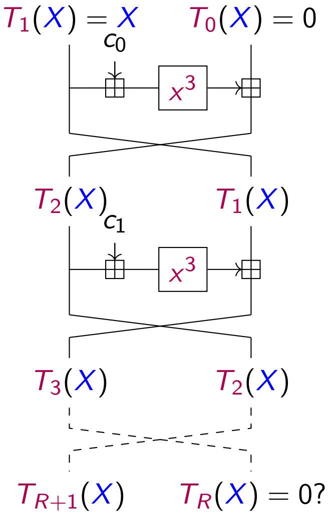
$$\begin{cases} T_0(X) = y_0 = 0 \\ T_1(X) = x_0 = X \\ T_{i+1}(X) = (T_i(X) + c_{i-1})^3 + T_{i-1}(X) \end{cases}$$



- 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}.



- 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}.
- Modelize the problem as $\{ T_R(X) = 0$.
- A solution $X = x$ gives a CICO solution: $(x, 0) \rightarrow (T_{R+1}(x), 0)$.

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, compute:

$$R = U \mod V$$

.
  - Invariant: If $R \neq 0$, $\gcd(U, V) = \gcd(V, R)$.
  - 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, compute:

$$R = U \mod V$$

.
  - Invariant: If $R \neq 0$, $\gcd(U, V) = \gcd(V, R)$.
  - Set $U, V = V, R$ and iterate.
  - If $R = 0$, return U.
- Apply the algorithm with $U = X^p - X$ and $V = T$ (degree $d \ll p$).

An improved first step of the Euclidian GCD algorithm

Goal: Compute $R = X^p - X \mod T$.

- The naive approach is too expensive ($p - d$ operations).
- Instead, 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 $R = X^p - X \mod T$.

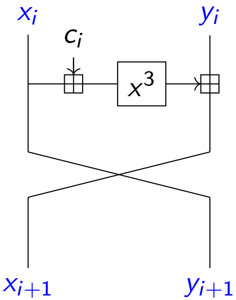
Root-finding Algorithm of a Polynomial in $\mathbb{F}_p$

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 $O(d \log(d) \log(p) \log(\log(d)))$ field operations
- Feasible in practice up to $d = 2^{32}$ (for $p \approx 2^{64}$).

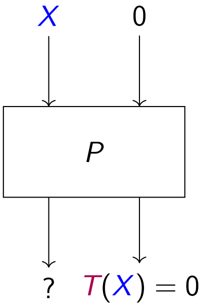
$\rightarrow$ We can break 21 rounds of Feistel-MiMC experimentally (out of 80 rounds).

Summary of the CICO cryptanalysis on Feistel-MiMC



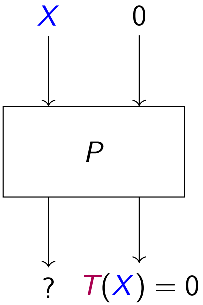
- Low degree round function (degree 3).

Summary of the CICO cryptanalysis on Feistel-MiMC



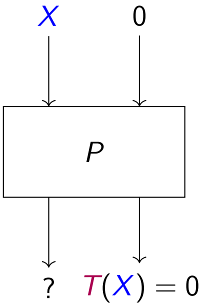
- Low degree round function (degree 3).
- Modelize CICO with a root-finding problem.

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 d ($O(d\log(d)\log(p))\log(\log(d))$).

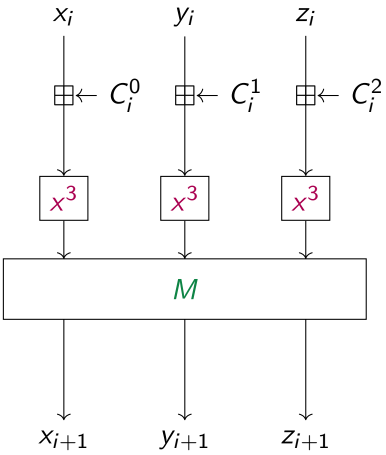
Summary of the CICO cryptanalysis on Feistel-MiMC



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

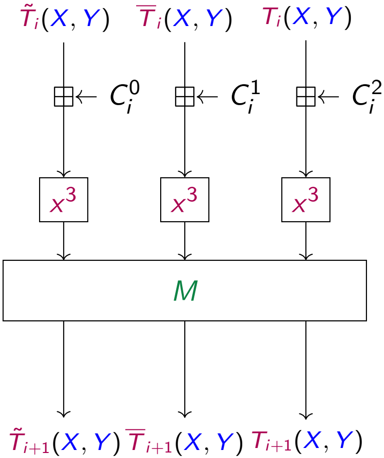
For a security level of 64 bits, 40 rounds are necessary.

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



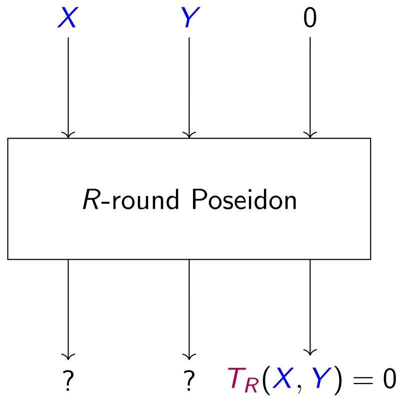
- Low degree round function.

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



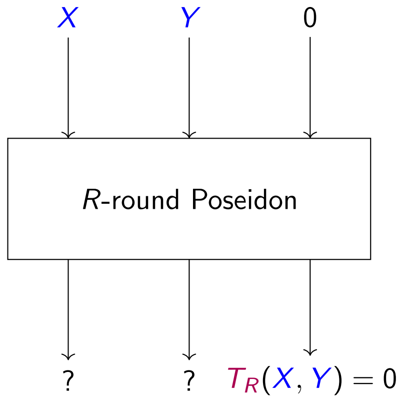
- Low degree round function.
- Set $\tilde{T}_0 = X = x_0, \overline{T}_0 = Y = y_0, T_0 = 0 = z_0$.
- Compute $\tilde{T}_i, \overline{T}_i, \tilde{T}_i$ for $i \leq R$.
- By induction, $T_i, \overline{T}_i, \tilde{T}_i$ are of degree 3^i.

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



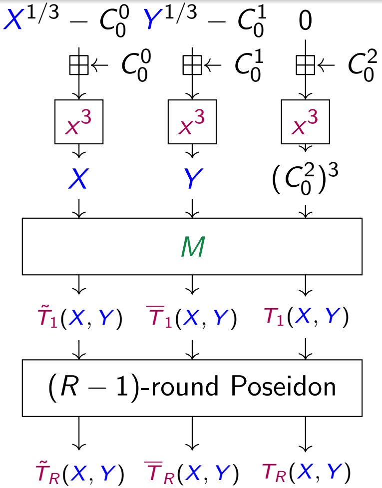
- Low degree round function.
- Set $\tilde{T}_0 = X = x_0$, $\overline{T}_0 = Y = y_0$, $T_0 = 0 = z_0$.
- Compute $\tilde{T}_i$, $\overline{T}_i$, $\tilde{T}_i$ for $i \leq R$.
- By induction, T_i, $\overline{T}_i$, $\tilde{T}_i$ are of degree 3^i.
- But $T_R(X, Y) = 0$ is underdetermined.
- Set Y to 0 and solve $T_R(X, 0) = 0$ (deg 3^R).

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



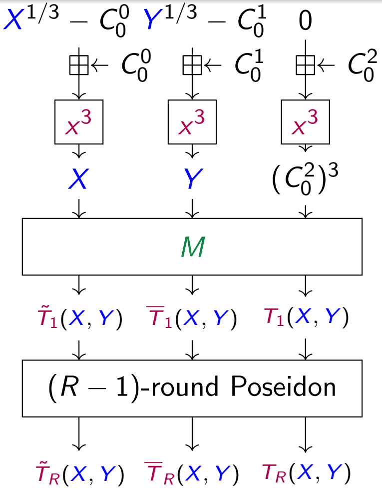
- Low degree round function.
- Set $\tilde{T}_0 = X = x_0, \overline{T}_0 = Y = y_0, T_0 = 0 = z_0$.
- Compute $\tilde{T}_i, \overline{T}_i, \tilde{T}_i$ for $i \leq R$.
- By induction, $T_i, \overline{T}_i, \tilde{T}_i$ are of degree 3^i.
- But $T_R(X, Y) = 0$ is underdetermined.
- Set Y to 0 and solve $T_R(X, 0) = 0$ (deg 3^R).
- Complexity: $O(R3^R)$.

The CICO Problem with Poseidon: Improvement



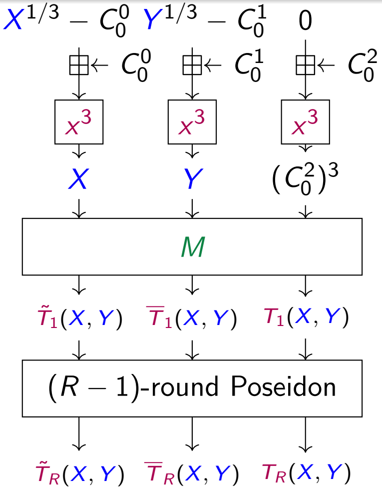
- Define X and Y after the first Sbox.

The CICO Problem with Poseidon: Improvement



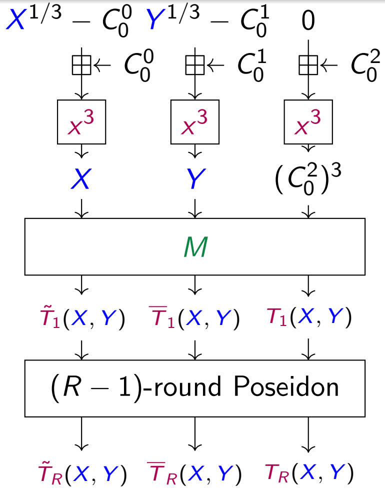
- Define X and Y after the first Sbox.
- By induction, T_i, $\overline{T}_i$, $\tilde{T}_i$ are of degree 3^{i-1}.

The CICO Problem with Poseidon: Improvement



- Define X and Y after the first Sbox.
- By induction, $T_i, \overline{T}_i, \tilde{T}_i$ are of degree 3^{i-1}.
- Set $Y = 0$ and solve $T_R(X, 0) = 0$ (deg 3^{R-1}).

The CICO Problem with Poseidon: Improvement

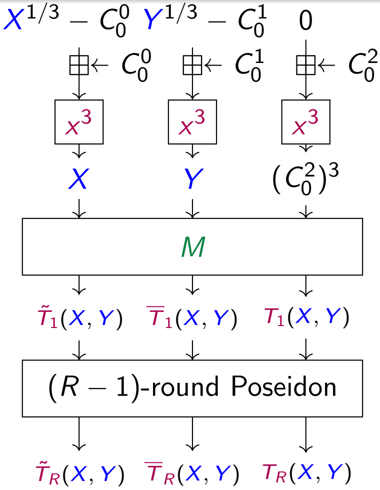


- Define X and Y after the first Sbox.
- By induction, T_i, $\overline{T}_i$, $\tilde{T}_i$ are of degree 3^{i-1}.
- Set $Y = 0$ and solve $T_R(X,0) = 0$ (deg 3^{R-1}).
- With a root x of T_R,

$$(x^{\frac{1}{3}} - C_0^0, -C_0^1, 0) \to (\tilde{T}_i(x,0), \overline{T}_i(x,0), 0)$$

is a CICO solution. (Recall: $x^{\frac{1}{3}} = x^{\frac{p-1}{3}}$)

The CICO Problem with Poseidon: Improvement



- Define X and Y after the first Sbox.
- By induction, $T_i, \overline{T}_i, \tilde{T}_i$ are of degree 3^{i-1}.
- Set $Y = 0$ and solve $T_R(X, 0) = 0$ (deg 3^{R-1}).
- With a root x of T_R,

$$(x^{\frac{1}{3}} - C_0^0, -C_0^1, 0) \to (\tilde{T}_i(x, 0), \overline{T}_i(x, 0), 0)$$

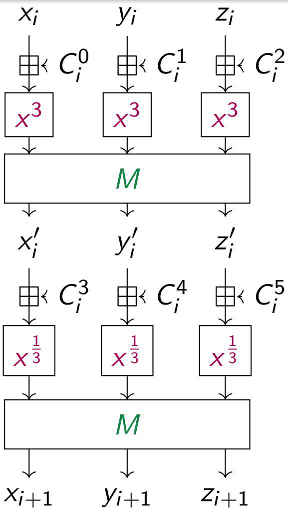
  is a CICO solution. (Recall: $x^{\frac{1}{3}} = x^{\frac{p-1}{3}}$)
- We bypass 1 round, thus gain a factor 3 in complexity.

The CICO Problem with low-degree round functions

For a round function of degree $\alpha \ll p$:

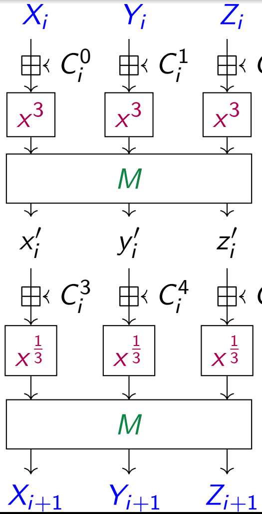
- Write the output as a polynomial of the input (degree α^r).
- Solve using univariate polynomial solving in $O(r\alpha^r)$ operations.
- For a security level of s, at least $s/\log_2(\alpha)$ rounds are needed.

The CICO Problem with Rescue Prime



- One SPN round with $S = x \to x^3$ and one SPN round with $S^{-1} = x \to x^{\frac{1}{3}} = x^{\frac{p-1}{3}}$.
- Problem: x_{i+1} can be written as a polynomial in x_i but of too high degree to solve.

The CICO Problem with Rescue Prime

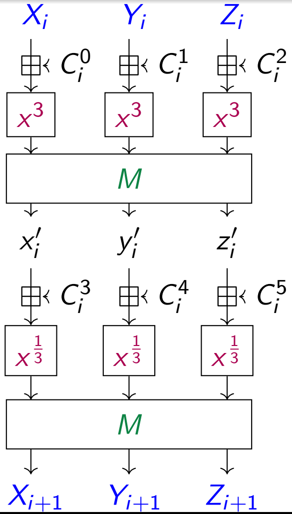


- One SPN round with $S = x \to x^3$ and one SPN round with $S^{-1} = x \to x^{\frac{1}{3}} = x^{\frac{p-1}{3}}$.
- Problem: x_{i+1} can be written as a polynomial in x_i but of too high degree to solve.
- Write $X_i = x_i, Y_i = y_i, Z_i = z_i$:

$$\begin{cases} x'i = T_{i,0}(X_i, Y_i, Z_i) \\ x'i = T_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) \end{cases}$$

- $T_{i,0}$ and $T_{i,1}$ are both of degree 3.

The CICO Problem with Rescue Prime



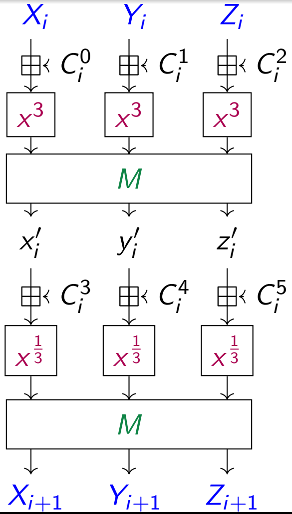
- One SPN round with $S = x \to x^3$ and one SPN round with $S^{-1} = x \to x^{\frac{1}{3}} = x^{\frac{p-1}{3}}$.
- Problem: x_{i+1} can be written as a polynomial in x_i but of too high degree to solve.
- Write $X_i = x_i, Y_i = y_i, Z_i = z_i$:

$$\begin{cases} x'i = T_{i,0}(X_i, Y_i, Z_i) \\ x'i = T_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) \end{cases}$$

- $T_{i,0}$ and $T_{i,1}$ are both of degree 3.

$\to$ $T_{i,0}(X_i, Y_i, Z_i) - T_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$

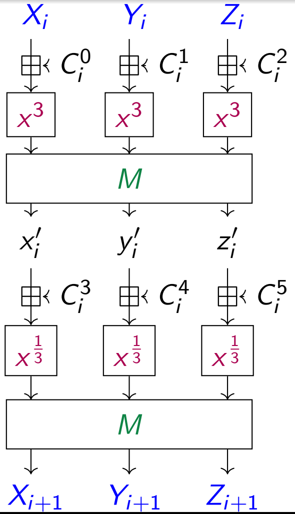
The CICO Problem with Rescue Prime



$$x'i \to \; T_{i,0}(X_i, Y_i, Z_i) - T_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$

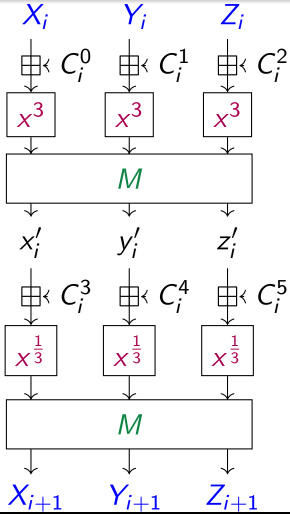
$$\begin{cases} y'i = \overline{T}_{i,0}(X_i, Y_i, Z_i) \\ y'i = \overline{T}_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) \end{cases}$$

The CICO Problem with Rescue Prime



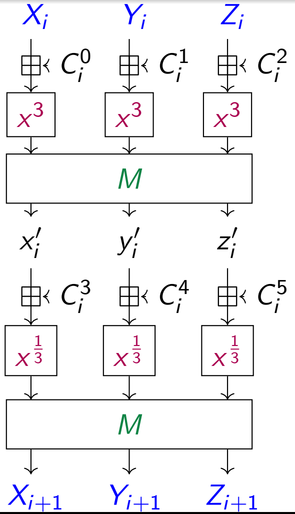
$$x'i \rightarrow T_{i,0}(X_i, Y_i, Z_i) - T_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$
$$y'i \rightarrow \overline{T}_{i,0}(X_i, Y_i, Z_i) - \overline{T}_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$

The CICO Problem with Rescue Prime



$$x'i \rightarrow T_{i,0}(X_i, Y_i, Z_i) - T_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$
$$y'i \rightarrow \overline{T}_{i,0}(X_i, Y_i, Z_i) - \overline{T}_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$
$$z'i \rightarrow \tilde{T}_{i,0}(X_i, Y_i, Z_i) - \tilde{T}_{i,1}(X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$

The CICO Problem with Rescue Prime



$$x'i \rightarrow T_i(X_i, Y_i, Z_i, X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$

$$y'i \rightarrow \overline{T}_i(X_i, Y_i, Z_i, X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$

$$z'i \rightarrow \tilde{T}_i(X_i, Y_i, Z_i, X_{i+1}, Y_{i+1}, Z_{i+1}) = 0$$

- R-round Rescue Prime: system of $3R$ equations:

$$\begin{cases} T_0(X_0, Y_0, Z_0, X_1, Y_1, Z_1) & = 0 \\ \overline{T}_0(X_0, Y_0, Z_0, X_1, Y_1, Z_1) & = 0 \\ \tilde{T}_0(X_0, Y_0, Z_0, X_1, Y_1, Z_1) & = 0 \\ \quad \vdots & \\ T_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, Z_R) & = 0 \\ \overline{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, Z_R) & = 0 \\ \tilde{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, Z_R) & = 0 \end{cases}$$

- R-round Rescue Prime: system of $3R$ equations:

$$\begin{cases} T_0(X_0, Y_0, 0, X_1, Y_1, Z_1) & = 0 \\ \overline{T}_0(X_0, Y_0, 0, X_1, Y_1, Z_1) & = 0 \\ \tilde{T}_0(X_0, Y_0, 0, X_1, Y_1, Z_1) & = 0 \\ \quad \vdots & \\ T_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0 \\ \overline{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0 \\ \tilde{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0 \end{cases}$$

- Input/Output Constraints: Replace Z_0 and Z_R with 0.

- R-round Rescue Prime: system of $3R$ equations:

$$\begin{cases} T_0(X_0, 0, 0, X_1, Y_1, Z_1) & = 0 \\ \overline{T}_0(X_0, 0, 0, X_1, Y_1, Z_1) & = 0 \\ \tilde{T}_0(X_0, 0, 0, X_1, Y_1, Z_1) & = 0 \\ \quad \vdots & \\ T_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0 \\ \overline{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0 \\ \tilde{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0 \end{cases}$$

- Input/Output Constraints: Replace Z_0 and Z_R with 0.
- $3R$ equations for $3R + 1$ variables: fix $Y_0 = 0$.

Solving the System with Groebner Basis

Full course needed to understand Groebner Basis (sometimes not enough).

a. **F5:** Compute a Groebner Basis of the system w.r.t the grevlex order.
   - Might be computationally expensive. The complexity highly depends on the regularity of the system.

b. **FGLM:** Convert it into a Groebner Basis w.r.t the lexical order.
   - Is in practice the most expensive step. Its complexity is well understood.

c. Retrieve the solutions from the lex Groebner Basis (cheap).

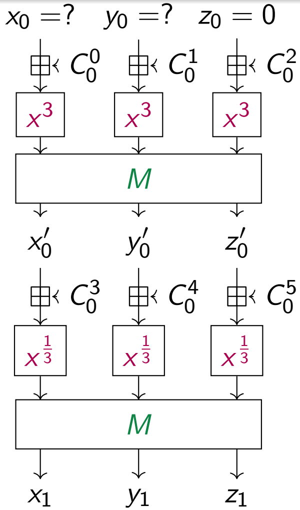
For simplicity, we will study the complexity of step b.

Rescue Prime Solving

- For n equations of degree d, step b. costs $O(nd^{n\omega})$, where $2 \leq \omega \leq 3$ is the matrix multiplication exponent.
- Rescue Prime: $3R$ equations of degree $3 \to O(3R3^{3R\omega})$.
- Breakable up to 3 round in practice.

We can actually do better by carefully analyzing the system.

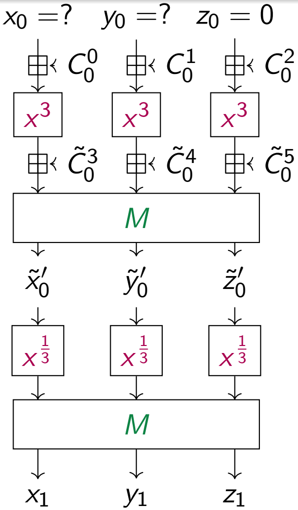
Rescue Prime: Bypass a round



- Swap M and the second constant addition:

$$\begin{pmatrix} \tilde{C}_0^3 \\ \tilde{C}_1^4 \\ \tilde{C}_2^5 \end{pmatrix} = M^{-1} \begin{pmatrix} C_0^3 \\ C_1^4 \\ C_2^5 \end{pmatrix} .$$

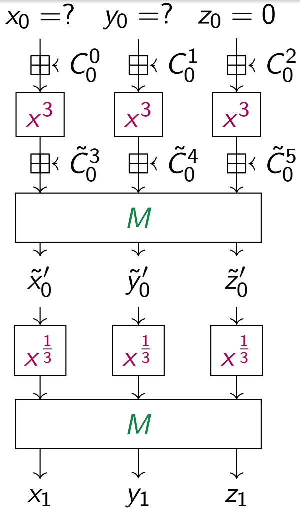
Rescue Prime: Bypass a round



- Swap M and the second constant addition:

$$\begin{pmatrix} \tilde{C}_0^3 \\ \tilde{C}_1^4 \\ \tilde{C}_2^5 \end{pmatrix} = M^{-1} \begin{pmatrix} C_0^3 \\ C_1^4 \\ C_2^5 \end{pmatrix} .$$

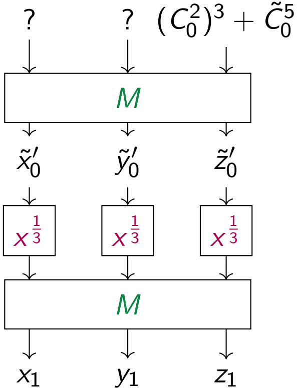
Rescue Prime: Bypass a round



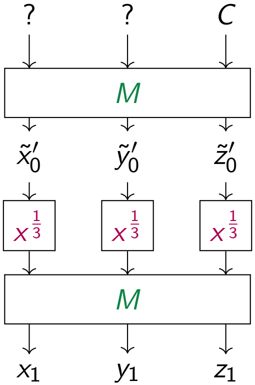
- Swap M and the second constant addition:
- Propagate the constaint $z_0 = 0$.

Rescue Prime: Bypass a round

- Swap M and the second constant addition:
- Propagate the constaint $z_0 = 0$.

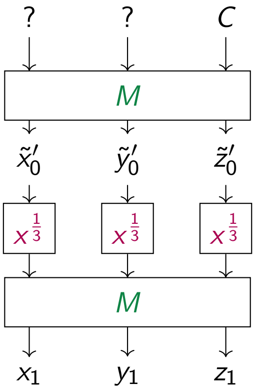


Rescue Prime: Bypass a round



- Swap M and the second constant addition:
- Propagate the constaint $z_0 = 0$.
- Call $C = (C_0^2)^3 + \tilde{C}_0^5$.

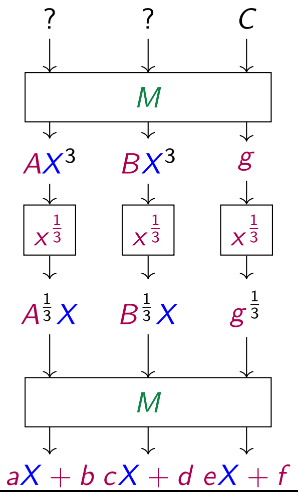
Rescue Prime: Bypass a round



- Swap M and the second constant addition:
- Propagate the constaint $z_0 = 0$.
- Call $C = (C_0^2)^3 + \tilde{C}_0^5$.
- Find g, A, B such that:

$$M^{-1}\begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ C \end{pmatrix}, M^{-1}\begin{pmatrix} A \\ B \\ 0 \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ 0 \end{pmatrix}.$$

Rescue Prime: Bypass a round



- Swap M and the second constant addition:
- Propagate the constaint $z_0 = 0$.
- Call $C = (C_0^2)^3 + \tilde{C}_0^5$.
- Find g, A, B such that:

$$M^{-1}\begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ C \end{pmatrix}, M^{-1}\begin{pmatrix} A \\ B \\ 0 \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ 0 \end{pmatrix}.$$

- This implies, for all X:

$$M^{-1}\begin{pmatrix} AX^3 \\ BX^3 \\ g \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ C \end{pmatrix}.$$

- R-round Rescue Prime: system of $3(R-1)$ equations:

$$\begin{cases} T_1(X_1, Y_1, Z_1, X_2, Y_2, Z_2) & = 0 \\ \overline{T}_1(X_1, Y_1, Z_1, X_2, Y_2, Z_2) & = 0 \\ \tilde{T}_1(X_1, Y_1, Z_1, X_2, Y_2, Z_2) & = 0 \\ \quad \vdots & \\ T_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, Z_R) & = 0 \\ \overline{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, Z_R) & = 0 \\ \tilde{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, Z_R) & = 0 \end{cases}$$

- R-round Rescue Prime: system of $3(R-1)$ equations:

$$
\begin{cases}
T_1(aX+b, cX+d, eX+f, X_2, Y_2, Z_2) & = 0 \\
\overline{T}_1(aX+b, cX+d, eX+f, X_2, Y_2, Z_2) & = 0 \\
\tilde{T}_1(aX+b, cX+d, eX+f, X_2, Y_2, Z_2) & = 0 \\
\quad\vdots & \\
T_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0 \\
\overline{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0 \\
\tilde{T}_{R-1}(X_{R-1}, Y_{R-1}, Z_{R-1}, X_R, Y_R, 0) & = 0
\end{cases}
$$

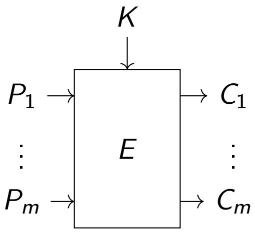
- Input Constraints: Replace X_1, Y_1, Z_1.
- Output Constraints: Replace Z_R with 0.

Improvement on Rescue-Prime

- From a solution X, we can build a CICO solution to Rescue-Prime.
- We improve our attack from 3 to 4 rounds of Rescue-Prime.
- Similarly, we can use this trick to bypass a 2nd round of Poseidon.

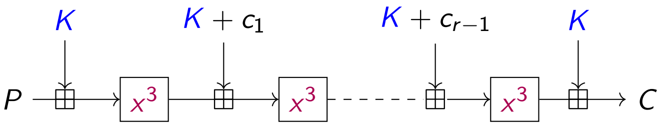
What now for key-recovery attacks?

Arithmetization-oriented block-ciphers



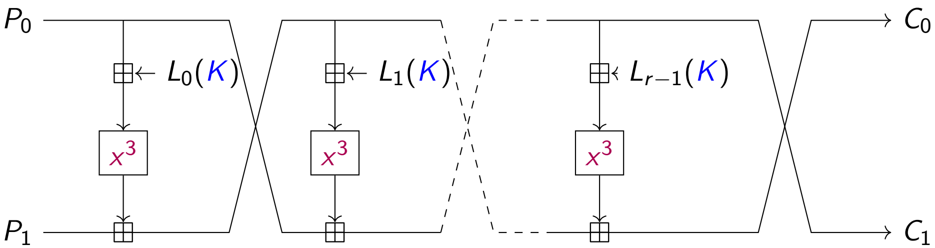
- Cipher $E : \mathbb{F}_p^m \times \mathbb{F}_p \to \mathbb{F}_p^m$.
- Goal: From captured plaintext/ciphertext pairs, recover K.

Interpolation attack on MiMC ($m = 1$)



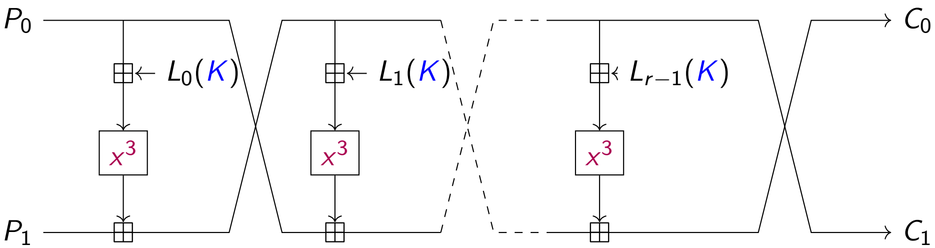
- Low degree Sbox: $x \to x^3$.
- $C = T_P(K)$, where T_P is of degree 3^r.
- Intercept a plaintext/ciphertext pair (P, C) and solve $T_P(K) - C$ in K.
- Complexity in $O(r \times 3^{3r})$.

Meet-In-The-Middle Interpolation attack on G-MiMC ($m = 2$)



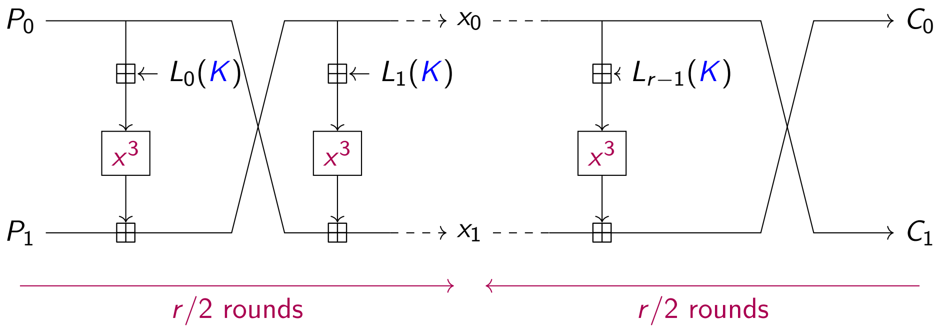
- Feistel network; L_i are linear functions of K.

Meet-In-The-Middle Interpolation attack on G-MiMC ($m = 2$)



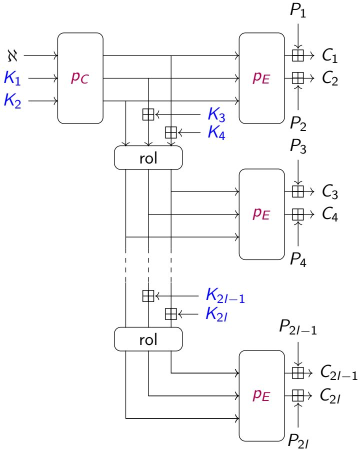
- Feistel network; L_i are linear functions of K.
- First idea: write $C_1 = T_{1,P_0,P_1}(K)$ and get a pair (P_0, P_1, C_0, C_1).
- Solve $T_{1,P_0,P_1}(K) - C_1 = 0$ in K (degree 3^{r-1}).

Meet-In-The-Middle Interpolation attack on G-MiMC ($m = 2$)

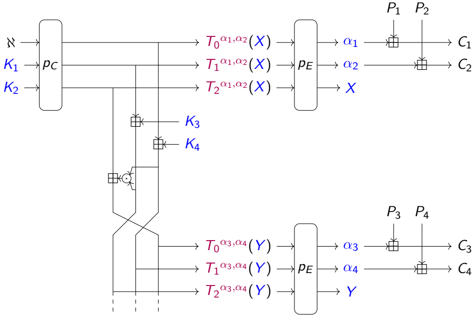


- Better idea: Exploit the low degree inverse round.
- $x_0 = T_{0,P_0,P_1}(K)$ and $x_0 = \overline{T}_{0,C_0,C_1}(K)$ of degrees $\sim 3^{r/2}$.
- Solve $T_{0,P_0,P_1}(K) - \overline{T}_{0,C_0,C_1}(K) = 0$ (degree $\sim 3^{r/2}$).

Ciminion



- K_i are key expansion the master key K.
- We interpret the K_i as independant subkeys.
- p_C of very high degree. p_E of reasonable degree (2^9 for $p \approx 2^{64}$).
- Goal: retrieve the intermediate state (before p_E), invert p_C and recover K_1, K_2.



- Ask for a ciphertext pair (P, C).
- Deduce $\alpha_1, \alpha_2, \alpha_3, \alpha_4$.
- Define output variables X and Y:

$$\begin{cases} T_0^{\alpha_1,\alpha_2}(X) = T_1^{\alpha_3,\alpha_4}(Y) - K_4 \\ T_1^{\alpha_1,\alpha_2}(X) = T_2^{\alpha_3,\alpha_4}(Y) - K_3 \\ T_2^{\alpha_1,\alpha_2}(X) = T_0^{\alpha_3,\alpha_4}(Y) \\ \qquad\qquad - T_1^{\alpha_3,\alpha_4}(Y) \odot T_2^{\alpha_3,\alpha_4}(Y) \end{cases}$$

- Problem: 3 equations for 4 unkown variables $(X, Y, K_3, K_4) \to$ insufficient.
- Idea: Get two pairs of plaintext/ciphertext.

Algebraic Attack on Ciminion

- Generate two sets of equations with two plaintext/ciphertext pairs.

$$\begin{cases} T_0{}^{\alpha_1,\alpha_2}(X) = T_1{}^{\alpha_3,\alpha_4}(Y) - K_4 \\ T_1{}^{\alpha_1,\alpha_2}(X) = T_2{}^{\alpha_3,\alpha_4}(Y) - K_3 \\ T_2{}^{\alpha_1,\alpha_2}(X) = T_0{}^{\alpha_3,\alpha_4}(Y) - T_1{}^{\alpha_3,\alpha_4}(Y) \odot T_2{}^{\alpha_3,\alpha_4}(Y) \\ T_0{}^{\alpha'_1,\alpha'_2}(X') = T_1{}^{\alpha'_3,\alpha'_4}(Y') - K_4 \\ T_1{}^{\alpha'_1,\alpha'_2}(X') = T_2{}^{\alpha'_3,\alpha'_4}(Y') - K_3 \\ T_2{}^{\alpha'_1,\alpha'_2}(X') = T_0{}^{\alpha'_3,\alpha'_4}(Y') - T_1{}^{\alpha'_3,\alpha'_4}(Y') \odot T_2{}^{\alpha'_3,\alpha'_4}(Y') \end{cases}$$

Algebraic Attack on Ciminion

- Generate two sets of equations with two plaintext/ciphertext pairs.
- Remove the variables (K_3, K_4):

$$\begin{cases} {T_0}^{\alpha_1,\alpha_2}(X) - {T_0}^{\alpha_1',\alpha_2'}(X') = {T_1}^{\alpha_3,\alpha_4}(Y) - {T_1}^{\alpha_3',\alpha_4'}(Y') \\ {T_1}^{\alpha_1,\alpha_2}(X) - {T_1}^{\alpha_1',\alpha_2'}(X') = {T_2}^{\alpha_3,\alpha_4}(Y) - {T_2}^{\alpha_3',\alpha_4'}(Y') \\ {T_2}^{\alpha_1,\alpha_2}(X) = {T_0}^{\alpha_3,\alpha_4}(Y) - {T_1}^{\alpha_3,\alpha_4}(Y)\,{T_2}^{\alpha_3,\alpha_4}(Y) \\ {T_2}^{\alpha_1',\alpha_2'}(X') = {T_0}^{\alpha_3',\alpha_4'}(Y') - {T_1}^{\alpha_3',\alpha_4'}(Y')\,{T_2}^{\alpha_3',\alpha_4'}(Y') \end{cases} .$$

- System of 4 equations of degree $\approx 2^9 \rightarrow$ Solve in $\approx 2^{36\omega}$ operations $(2 \leq \omega \leq 3)$, for $p \approx 2^{64}$.

Algebraic Attack on Ciminion

- Generate two sets of equations with two plaintext/ciphertext pairs.
- Remove the variables (K_3, K_4):

$$\begin{cases} {T_0}^{\alpha_1,\alpha_2}(X) - {T_0}^{\alpha_1',\alpha_2'}(X') = {T_1}^{\alpha_3,\alpha_4}(Y) - {T_1}^{\alpha_3',\alpha_4'}(Y') \\ {T_1}^{\alpha_1,\alpha_2}(X) - {T_1}^{\alpha_1',\alpha_2'}(X') = {T_2}^{\alpha_3,\alpha_4}(Y) - {T_2}^{\alpha_3',\alpha_4'}(Y') \\ {T_2}^{\alpha_1,\alpha_2}(X) = {T_0}^{\alpha_3,\alpha_4}(Y) - {T_1}^{\alpha_3,\alpha_4}(Y){T_2}^{\alpha_3,\alpha_4}(Y) \\ {T_2}^{\alpha_1',\alpha_2'}(X') = {T_0}^{\alpha_3',\alpha_4'}(Y') - {T_1}^{\alpha_3',\alpha_4'}(Y'){T_2}^{\alpha_3',\alpha_4'}(Y') . \end{cases}$$

- System of 4 equations of degree $\approx 2^9 \rightarrow$ Solve in $\approx 2^{36\omega}$ operations $(2 \leq \omega \leq 3)$, for $p \approx 2^{64}$.
- This does not threaten the cipher, but has been missed out by the designers.

Conclusion

- We study public permutations and ciphers on big fields.
- Algebraic Cryptanalysis:
  - Modelize the system with a system of polynomial equations.
  - Solve it using Polynomial Root Finding or Groebner Basis.
- We estimate the complexity of the attack.
- We deduce a lower bound on the number of rounds for a given security level.

To go deeper: `https://tosc.iacr.org/index.php/ToSC/article/view/9850`

Conclusion

- We study public permutations and ciphers on big fields.
- Algebraic Cryptanalysis:
  - Modelize the system with a system of polynomial equations.
  - Solve it using Polynomial Root Finding or Groebner Basis.
- We estimate the complexity of the attack.
- We deduce a lower bound on the number of rounds for a given security level.

To go deeper: `https://tosc.iacr.org/index.php/ToSC/article/view/9850`

Thank you for your attention.