

Solving Polynomial Systems with Resultants to break Zero-Knowledge-Friendly Hash Functions

Augustin Bariant¹, Aurélien Boeuf², Pierre Briaud³, Maël Hostettler⁴,
Morten Øygarden³, and Håvard Raddum³

¹ANSSI, Paris, France

²INRIA, Paris, France

³Simula UiB, Bergen, Norway

⁴Télécom SudParis, Évry-Courcouronnes, France

Versailles Seminar, October 30th 2025

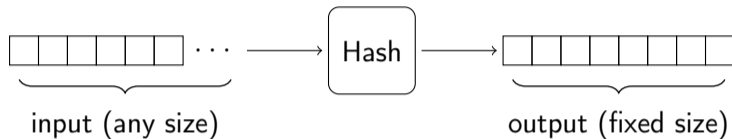
1 Hash Functions

2 Zero-Knowledge and Arithmetization

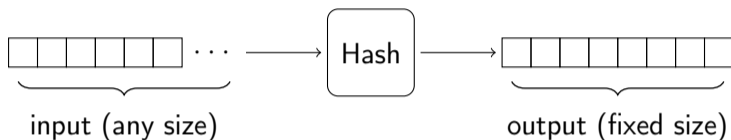
3 Resultants

4 Resultant-based Attacks

Hash functions



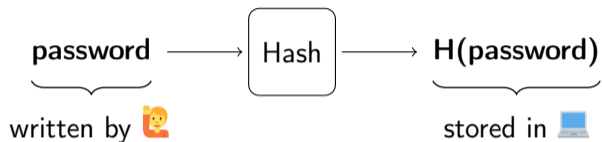
Hash functions



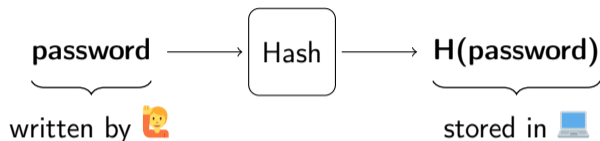
Cryptographically secure:

- **Collision resistance:** cannot find x_1, x_2 s.t. $H(x_1) = H(x_2)$.
- **Preimage resistance:** given y , cannot find x s.t. $H(x) = y$.
- **2nd Preimage resistance:** given $H(x_1)$, cannot find x_2 s.t. $H(x_1) = H(x_2)$.

Password-based PC authentication

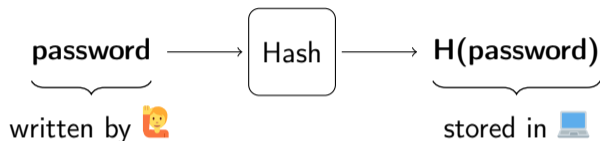


Password-based PC authentication



- If 🕵️ reads $H(\text{password})$ from 🖥️'s memory: no problem ✅
- Why?

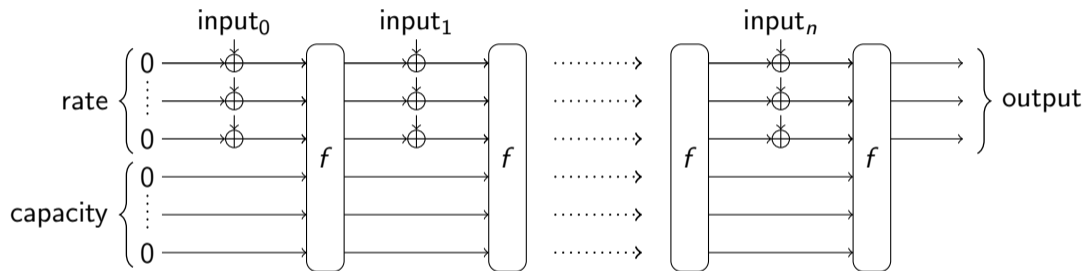
Password-based PC authentication



- If 🕵️ reads $H(\text{password})$ from 🖥️'s memory: no problem ✅
- Why? Preimage resistance!

$H(\text{password}) \not\rightarrow \text{password}.$

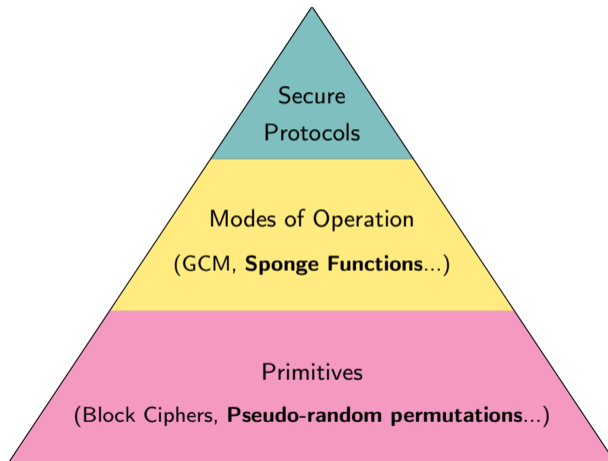
How to design secure hash functions?



The Sponge Construction

“Good” permutation $f \rightarrow$ Cryptographically secure Hash Function.

Cryptography in a nutshell




Online authentication



Online authentication



Better protocols exist, like OAuth 2.0,
but even in those, you have to
trust  with your **password**.

1 Hash Functions

2 Zero-Knowledge and Arithmetization

3 Resultants

4 Resultant-based Attacks

Zero-Knowledge Protocols

Goal: if $H(x) = y$ with x secret and y public, prove that you know x without disclosing it.

Zero-Knowledge Protocols

Goal: if $H(x) = y$ with x secret and y public, prove that you know x without disclosing it.

Is this kind of thing even possible?
Let's do a simple experiment!

Zero-Knowledge Protocols

Goal: if $H(x) = y$ with x secret and y public, prove that you know x without disclosing it.

Is this kind of thing even possible?
Let's do a simple experiment!

Issue: Arithmetically complex $H \implies$ Big, costly proof 📦

Arithmetization-Oriented Primitives

AOP = primitive with “simple” arithmetic representation

Classic	Arithmetization-Oriented
Binary Operations	Arithmetic Operations
Algebraically complex	Algebraically simple
Small field (\mathbb{F}_{2^8})	Large field ($\mathbb{F}_q, q > 2^{30}$)
e.g. AES, SHA-3	e.g. Poseidon, Rescue

Arithmetization for Zero-Knowledge

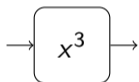
- Implemented using “constraint systems” (R1CS, AIR, Plonk...).
- **Less constraints + Low-degree = Better performance.**

Function \rightarrow Arithmetic Circuit \rightarrow Constraint System

Arithmetization for Zero-Knowledge

- Implemented using “constraint systems” (R1CS, AIR, Plonk...).
- **Less constraints + Low-degree = Better performance.**

Function \rightarrow Arithmetic Circuit \rightarrow Constraint System



Security: ✗ Low degree

Efficiency: ✓ Few constraints



Security: ✓ High degree

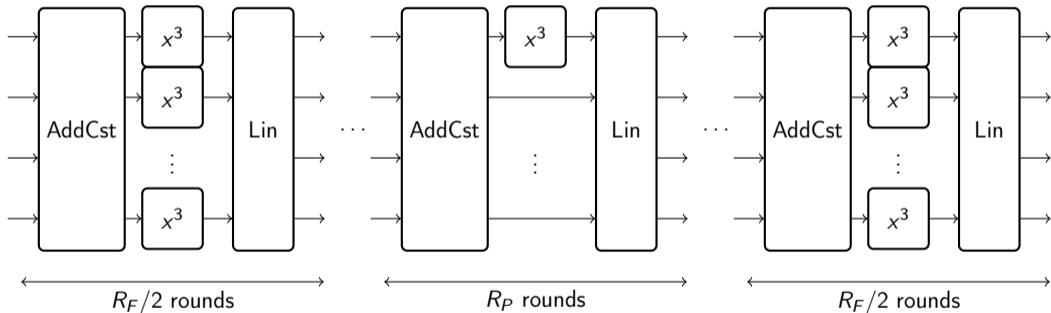
Efficiency: ✗ Many constraints



Security: ✓ High degree

Efficiency: ✓ Few constraints
(bc low degree inverse)
($x^{1/3} = y$ vs $y^3 = x$)

Example: Poseidon Permutation



1 Hash Functions

2 Zero-Knowledge and Arithmetization

3 Resultants

4 Resultant-based Attacks

Where do resultants come from?

Problem: Given polynomials $V(x)$ and $Q(x)$, do they have a common root (x' s.t. $V(x') = Q(x') = 0$?)

Where do resultants come from?

Problem: Given polynomials $V(x)$ and $Q(x)$, do they have a common root (x' s.t. $V(x') = Q(x') = 0$?)

- **Algorithmic answer:** Compute the roots of V and Q , check their intersection.
Advantage: **very efficient** for this specific problem. ✓

Where do resultants come from?

Problem: Given polynomials $V(x)$ and $Q(x)$, do they have a common root (x' s.t. $V(x') = Q(x') = 0$?)

- **Algorithmic answer:** Compute the roots of V and Q , check their intersection.

Advantage: **very efficient** for this specific problem. ✓

- **Algebraic answer:** Compute the **resultant** of V and Q !

Advantage: resultants are more **general** and **powerful**. ✓

$\text{Res}(V, Q) = 0$ if and only if V and Q have a common root.

Defining resultants

$\underbrace{V(x)}_{\text{degree } n}$ and $\underbrace{Q(x)}_{\text{degree } m}$ have a common root.

\Downarrow
 $\exists a$ such that $V = (x - a)S$ and $Q = (x - a)U$.

Defining resultants

$\underbrace{V(x)}_{\text{degree } n}$ and $\underbrace{Q(x)}_{\text{degree } m}$ have a common root.

\Downarrow
 $\exists a$ such that $V = (x - a)S$ and $Q = (x - a)U$.

$$\Downarrow$$
$$UV = SQ.$$

Defining resultants

$\underbrace{V(x)}_{\text{degree } n}$ and $\underbrace{Q(x)}_{\text{degree } m}$ have a common root.

$\exists a$ such that $V = (x - a)S$ and $Q = (x - a)U$.

$\left\{ \begin{array}{l} \exists S \text{ of degree } < n, \\ \exists U \text{ of degree } < m, \end{array} \right.$ such that $UV = SQ$.

Defining resultants

$\underbrace{V(x)}_{\text{degree } n}$ and $\underbrace{Q(x)}_{\text{degree } m}$ have a common root.

$\exists a$ such that $V = (x - a)S$ and $Q = (x - a)U$.

$\left\{ \begin{array}{l} \exists S \text{ of degree } < n, \\ \exists U \text{ of degree } < m, \end{array} \right.$ such that $UV = SQ$.

$\left\{ \begin{array}{l} \exists s_0, \dots, s_{n-1} \in \mathbb{F}_p, \\ \exists u_0, \dots, u_{m-1} \in \mathbb{F}_p, \end{array} \right.$ such that $\sum_{i=0}^{m-1} u_i(x^i V) = \sum_{j=0}^{n-1} s_j(x^j Q)$.

Defining resultants

$\underbrace{V(x)}_{\text{degree } n}$ and $\underbrace{Q(x)}_{\text{degree } m}$ have a common root.

$\exists a$ such that $V = (x - a)S$ and $Q = (x - a)U$.

$\left\{ \begin{array}{l} \exists S \text{ of degree } < n, \\ \exists U \text{ of degree } < m, \end{array} \right.$ such that $UV = SQ$.

$\left\{ \begin{array}{l} \exists s_0, \dots, s_{n-1} \in \mathbb{F}_p, \\ \exists u_0, \dots, u_{m-1} \in \mathbb{F}_p, \end{array} \right.$ such that $\sum_{i=0}^{m-1} u_i (x^i V) = \sum_{j=0}^{n-1} s_j (x^j Q)$.

Linear algebra!

The Sylvester Matrix

$\underbrace{V(x)}_{\text{degree } n}$ and $\underbrace{Q(x)}_{\text{degree } m}$ have a common root.

$(V, \dots, x^{m-1}V, Q, \dots, x^{n-1}Q)$ are linearly dependent.

The Sylvester Matrix

$\underbrace{V(x)}_{\text{degree } n}$ and $\underbrace{Q(x)}_{\text{degree } m}$ have a common root.

$(V, \dots, x^{m-1}V, Q, \dots, x^{n-1}Q)$ are linearly dependent.

$$\text{Res}(V, Q) = \det \begin{bmatrix} v_0 & \cdots & v_n & & 0 \\ & \ddots & & \ddots & \\ 0 & & v_0 & \cdots & v_n \\ q_0 & \cdots & q_m & & 0 \\ & \ddots & & \ddots & \\ 0 & & q_0 & \cdots & q_m \end{bmatrix} = 0.$$

} m
} n
} $n+m$

Resultant of multivariate polynomials

Problem: Given $V(x, y)$ and $Q(x, y)$, do x', y' exist s.t. $V(x', y') = Q(x', y') = 0$?

Resultant of multivariate polynomials

Problem: Given $V(x, y)$ and $Q(x, y)$, do x', y' exist s.t. $V(x', y') = Q(x', y') = 0$?

Solution: Compute a **univariate polynomial** from V and Q , i.e. **eliminate** x or y .

Resultant of multivariate polynomials

Problem: Given $V(x, y)$ and $Q(x, y)$, do x', y' exist s.t. $V(x', y') = Q(x', y') = 0$?

Solution: Compute a **univariate polynomial** from V and Q , i.e. **eliminate** x or y .

- **Gröbner bases:** Compute a GB of $\langle V, Q \rangle$ (costly 💰), change its order to lex (costly 💰) to get a **univariate polynomial**.

Resultant of multivariate polynomials

Problem: Given $V(x, y)$ and $Q(x, y)$, do x', y' exist s.t. $V(x', y') = Q(x', y') = 0$?

Solution: Compute a **univariate polynomial** from V and Q , i.e. **eliminate** x or y .

- **Gröbner bases:** Compute a GB of $\langle V, Q \rangle$ (costly 💰), change its order to lex (costly 💰) to get a **univariate polynomial**.
- **Resultants:** See V and Q as “univariate polynomials” in $(\mathbb{F}_p[x])[y]$:

$\text{Res}_y(V, Q)$ is a **univariate polynomial** $\in \mathbb{F}_p[x]$!

Why does it work?

Problem: Given $V(x, y)$ and $Q(x, y)$, do x', y' exist s.t. $V(x', y') = Q(x', y') = 0$?

Why does it work?

Problem: Given $V(x, y)$ and $Q(x, y)$, do x', y' exist s.t. $V(x', y') = Q(x', y') = 0$?

- For all $x' \in \mathbb{F}_p$, define $R(x') = \text{Res}(\underbrace{V(x', y)}_{\in \mathbb{F}_p[y]}, \underbrace{Q(x', y)}_{\in \mathbb{F}_p[y]})$.
- There exists $y' \in \mathbb{F}_p$ s.t. $V(x', y') = Q(x', y') = 0$ iff $R(x') = 0$.

Why does it work?

Problem: Given $V(x, y)$ and $Q(x, y)$, do x', y' exist s.t. $V(x', y') = Q(x', y') = 0$?

- For all $x' \in \mathbb{F}_p$, define $R(x') = \text{Res}(\underbrace{V(x', y)}_{\in \mathbb{F}_p[y]}, \underbrace{Q(x', y)}_{\in \mathbb{F}_p[y]})$.
- There exists $y' \in \mathbb{F}_p$ s.t. $V(x', y') = Q(x', y') = 0$ iff $R(x') = 0$.
- R is defined as $\text{Res}_y(V, Q)$: **univariate polynomial in x** .
- **Variable elimination:** If V, Q are in x_1, \dots, x_k , $\text{Res}_{x_i}(V, Q)$ has no x_i .

1 Hash Functions

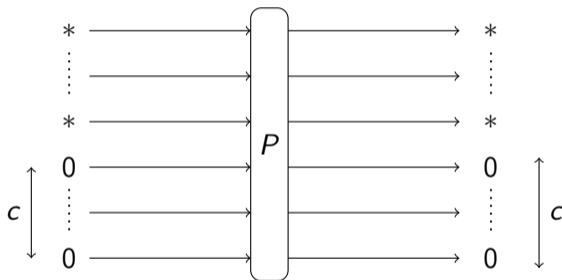
2 Zero-Knowledge and Arithmetization

3 Resultants

4 Resultant-based Attacks

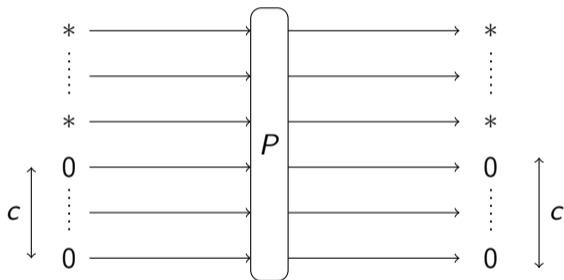
Attacking Permutations: The CICO Problem

CICO- c problem for P : find an input/output pair s.t.



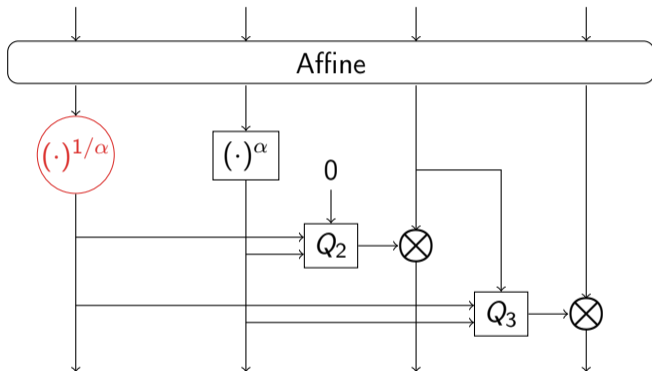
Attacking Permutations: The CICO Problem

CICO- c problem for P : find an input/output pair s.t.

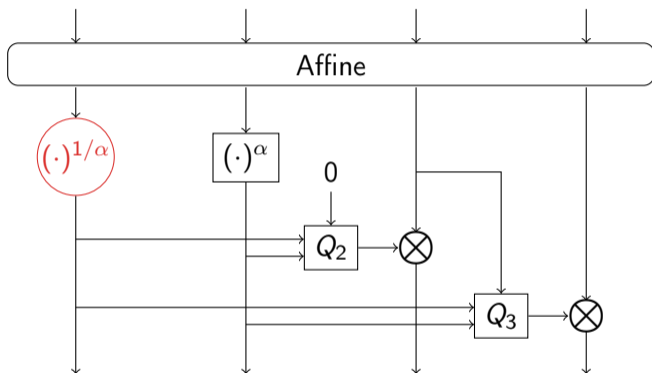


- PoSSo attack: find a CICO solution by solving a multivariate model of P .

Example: Modelling Griffin



Example: Modelling Griffin



$(\cdot)^{1/\alpha}$ is the only high degree operation \rightarrow add a new variable for each $(\cdot)^{1/\alpha}$.

The Models in CICO-1

$$\left\{ \begin{array}{l} V_1(x_1, x_2) = 0, \\ V_2(x_1, x_2, x_3) = 0, \\ \vdots \\ V_{k-1}(x_1, \dots, x_k) = 0, \\ V_k(x_1, \dots, x_k) = 0. \end{array} \right.$$

“Quasi-triangular” system!

The Attack

Resultant in $x_i \implies$ elimination of x_i .

The Attack

Resultant in $x_i \implies$ elimination of x_i .

- $V_{k-1}^* \leftarrow \text{Res}_{x_k}(V_{k-1}, V_k) \implies$ eliminated x_k everywhere! ✓

The Attack

Resultant in $x_i \implies$ elimination of x_i .

- $V_{k-1}^* \leftarrow \text{Res}_{x_k}(V_{k-1}, V_k) \implies$ eliminated x_k everywhere! ✓
- $V_{k-2}^* \leftarrow \text{Res}_{x_{k-1}}(V_{k-2}, V_{k-1}^*) \implies$ eliminated x_{k-1} everywhere! ✓
- ...
- $V_1^* \leftarrow \text{Res}_{x_2}(V_1, V_2^*) \implies$ univariate in x_1 ! ✓

The Attack

Resultant in $x_i \implies$ elimination of x_i .

- $V_{k-1}^* \leftarrow \text{Res}_{x_k}(V_{k-1}, V_k) \implies$ eliminated x_k everywhere! ✓
- $V_{k-2}^* \leftarrow \text{Res}_{x_{k-1}}(V_{k-2}, V_{k-1}^*) \implies$ eliminated x_{k-1} everywhere! ✓
- ...
- $V_1^* \leftarrow \text{Res}_{x_2}(V_1, V_2^*) \implies$ univariate in x_1 ! ✓

Quasi-triangular system \implies degree of V_1^* is minimal. 🍌

In general, Resultants are...

$$\left\{ \begin{array}{l} V_1(x_1, \dots, x_k) = 0 \text{ (degree } d), \\ V_2(x_1, \dots, x_k) = 0 \text{ (degree } d), \\ \vdots \\ V_{k-1}(x_1, \dots, x_k) = 0 \text{ (degree } d), \\ V_k(x_1, \dots, x_k) = 0 \text{ (degree } d). \end{array} \right.$$

- Minimal univariate degree (D_I) is at most d^k .
- $\text{Res}_{x_i}(V_j, V_{j'})$ has degree at most d^2 .
- “Gaussian elimination” with Resultants ends at degree at most $d^{2^{k-1}}$!

The results

	Griffin	Anemoi	Arion	Rescue (security 2^{512})
Freelunch (last step, GB-based)	2^{64}	2^{118}	2^{83}	-
Resultants	2^{55}	2^{79}	2^{68}	2^{475}

The results

	Griffin	Anemoi	Arion	Rescue (security 2^{512})
Freelunch (last step, GB-based)	2^{64}	2^{118}	2^{83}	-
Resultants	2^{55}	2^{79}	2^{68}	2^{475}

- Freelunch: $\mathcal{O}(D_I \alpha^{n(\omega-1)})$,
- Resultants: $\mathcal{O}(D_I \alpha^2 2^n)$.

($D_I = \min.$ univariate degree, $n = \text{round number}$,
 $\alpha \in \{3, 5, 7\}$, $2 \leq \omega < 3$).

Conclusion

- Gröbner bases are not always the best tools.
- Resultants have surprising uses.
- D_l^ω (FGLM lower bound) should not be taken as a security goal.

Conclusion

- Gröbner bases are not always the best tools.
- Resultants have surprising uses.
- D_l^ω (FGLM lower bound) should not be taken as a security goal.

Use a password manager!