Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC

Martin Albrecht – Carlos Cid – Lorenzo Grassi – Dmitry Khovratovich – **Reinhard Lüftenegger** – Christian Rechberger – Markus Schofnegger

Asiacrypt 2019

SCIENCE
PASSION
TECHNOLOGY

To Put the Cart Before the Horse...

Our main contribution is a known-plaintext key-recovery attack on the block cipher JARVIS with a **single** plaintext-ciphertext pair.

Rounds	Security level (bits)	Attack complexity ($\log_2 \#ops$)
10 (JARVIS-128)	128	**72**
12 (JARVIS-192)	192	**85**
14 (JARVIS-256)	256	**98**

- Practically verified up to 6 rounds of JARVIS
- Extends to a preimage attack on the hash function FRIDAY

To Put the Cart Before the Horse...

Our main contribution is a known-plaintext key-recovery attack on the block cipher JARVIS with a **single** plaintext-ciphertext pair.

Rounds	Security level (bits)	Attack complexity ($\log_2 \#ops$)
10 (JARVIS-128)	128	**72**
12 (JARVIS-192)	192	**85**
14 (JARVIS-256)	256	**98**

- Practically verified up to 6 rounds of JARVIS
- Extends to a preimage attack on the hash function FRIDAY

Overview

Introduction

- Preliminaries
- The MARVELlous Design

Key-Recovery Attack on JARVIS

- Attack Idea
- Results

Algebraic Cryptanalysis

- **Model** a cryptographic primitive as a system of multivariate polynomial equations

$$f_1(x_1, \ldots, x_n) = \cdots = f_k(x_1, \ldots, x_n) = 0$$

in several variables $x_1, \ldots, x_n$ over some finite field $\mathbb{F} \longrightarrow$ In general, result is a **non-linear** equation system

- **Solve** the system (e.g. for a specific variable) $\longrightarrow$ Several techniques available. **Gröbner bases** are one of them.

Algebraic Cryptanalysis

- **Model** a cryptographic primitive as a system of multivariate polynomial equations

$$f_1(x_1, \ldots, x_n) = \cdots = f_k(x_1, \ldots, x_n) = 0$$

  in several variables $x_1, \ldots, x_n$ over some finite field $\mathbb{F} \longrightarrow$ In general, result is a **non-linear** equation system

- **Solve** the system (e.g. for a specific variable) $\longrightarrow$ Several techniques available. **Gröbner bases** are one of them.

Solving Equation Systems with Gröbner Bases

- Formally, a Gröbner basis is a **special generating set** for an ideal in a multivariate polynomial ring
- Informally, a Gröbner basis is a **different representation** of an equation system with the same solution set
- Gröbner bases assist in solving systems of polynomial equations over some (finite) field $\mathbb{F}$
- Used together with factorisation algorithms for **univariate** polynomials

Solving Equation Systems with Gröbner Bases

- Formally, a Gröbner basis is a **special generating set** for an ideal in a multivariate polynomial ring
- Informally, a Gröbner basis is a **different representation** of an equation system with the same solution set
- Gröbner bases assist in solving systems of polynomial equations over some (finite) field $\mathbb{F}$
- Used together with factorisation algorithms for **univariate** polynomials

Solving Equation Systems with Gröbner Bases

- Formally, a Gröbner basis is a **special generating set** for an ideal in a multivariate polynomial ring
- Informally, a Gröbner basis is a **different representation** of an equation system with the same solution set
- Gröbner bases assist in solving systems of polynomial equations over some (finite) field $\mathbb{F}$
- Used together with factorisation algorithms for **univariate** polynomials

Solving Equation Systems with Gröbner Bases

- Formally, a Gröbner basis is a **special generating set** for an ideal in a multivariate polynomial ring
- Informally, a Gröbner basis is a **different representation** of an equation system with the same solution set
- Gröbner bases assist in solving systems of polynomial equations over some (finite) field $\mathbb{F}$
- Used together with factorisation algorithms for **univariate** polynomials

MARVELlous

- MARVELlous [AD18] is a family of cryptographic primitives, comprising **Jarvis** (block cipher) and **Friday** (hash function)
- Designed to be efficient in the **STARK** setting
- "Algebraic" design that works with low-degree polynomials
- The hash function Friday is based on the block cipher Jarvis

MARVELlous

- MARVELlous [AD18] is a family of cryptographic primitives, comprising **Jarvis** (block cipher) and **Friday** (hash function)
- Designed to be efficient in the **STARK** setting
- “Algebraic” design that works with low-degree polynomials
- The hash function Friday is based on the block cipher Jarvis

MARVELlous

- MARVELlous [AD18] is a family of cryptographic primitives, comprising **JARVIS** (block cipher) and **FRIDAY** (hash function)
- Designed to be efficient in the **STARK** setting
- “Algebraic” design that works with low-degree polynomials
- The hash function FRIDAY is based on the block cipher JARVIS

MARVELlous

- MARVELlous [AD18] is a family of cryptographic primitives, comprising **Jarvis** (block cipher) and **Friday** (hash function)
- Designed to be efficient in the **STARK** setting
- “Algebraic” design that works with low-degree polynomials
- The hash function Friday is based on the block cipher Jarvis

STARKs

STARK [BBH+18]
Scalable **T**ransparent **AR**gument of **K**nowledge

General goal: Given a public function f, a private input x and a public value y proof that $f(x) = y$ **without** revealing x.

Features of STARKs

- Arithmetisation-based
- Use Merkle-trees

$\longrightarrow$ requirement of dedicated hash-function designs for efficiency

STARKs

STARK [BBH+18]
Scalable **T**ransparent **AR**gument of **K**nowledge

General goal: Given a public function f, a private input x and a public value y proof that $f(x) = y$ **without** revealing x.

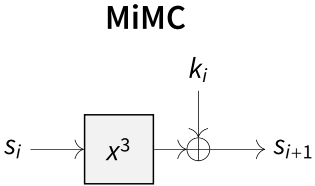
Features of STARKs

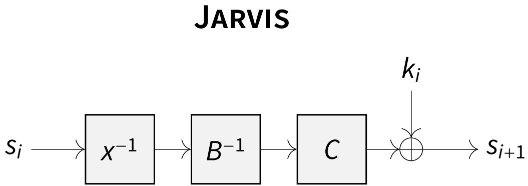
- Arithmetisation-based
- Use Merkle-trees

$\longrightarrow$ requirement of dedicated hash-function designs for efficiency

Jarvis: the Design

- Jarvis is similar to MiMC [AGR+16] and works entirely over $\mathbb{F}_{2^n}$, with $n \in \{128, 160, 192, 256\}$





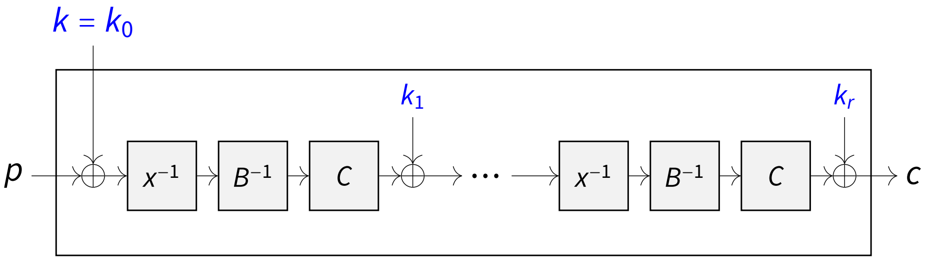
- B, C are affine polynomials of degree 4 and B^{-1} the compositional inverse of B.

Key-Recovery Attack on JARVIS I

Goal: Given **one** plaintext p and corresponding ciphertext $c = E_k(p)$ recover the secret key k.

Idea: Relate consecutive rounds by low-degree polynomial relations!

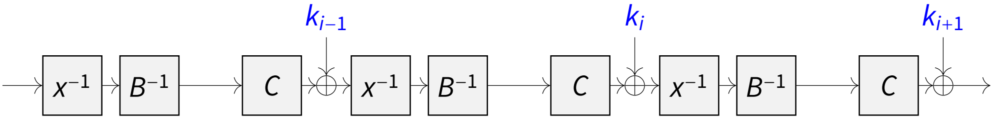
Key-Recovery Attack on JARVIS I



Goal: Given **one** plaintext p and corresponding ciphertext $c = E_k(p)$ recover the secret key k.

Idea: Relate consecutive rounds by low-degree polynomial relations!

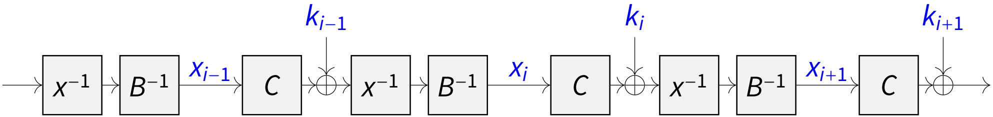
Key-Recovery Attack on JARVIS II



Basic strategy

- Introduce variables x_i for intermediate states between B^{-1} and C in each round
- Relate each x_i to the previous and next intermediate state x_{i-1} and x_{i+1} respectively

Key-Recovery Attack on Jarvis II

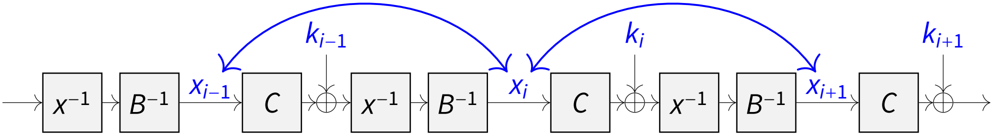


Basic strategy

- Introduce variables x_i for intermediate states between B^{-1} and C in each round

- Relate each x_i to the previous and next intermediate state x_{i-1} and x_{i+1} respectively

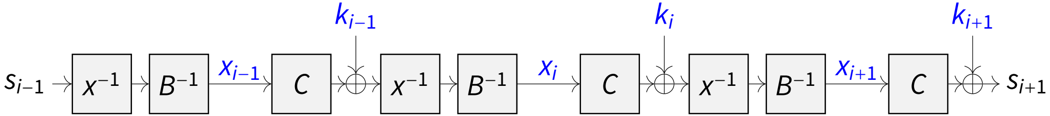
Key-Recovery Attack on JARVIS II



Basic strategy

- Introduce variables x_i for intermediate states between B^{-1} and C in each round
- Relate each x_i to the previous and next intermediate state x_{i-1} and x_{i+1} respectively

Key-Recovery Attack on JARVIS III

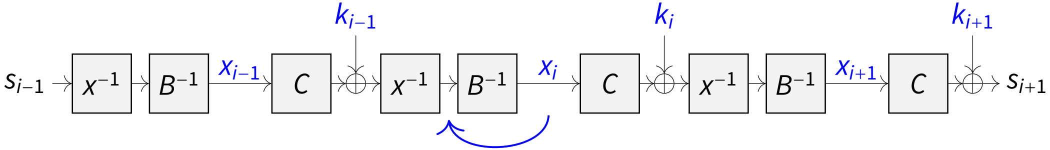


Basic equations

$$B(x_i) = \frac{1}{C(x_{i-1}) + k_{i-1}}$$

$$C(x_i) = \frac{1}{B(x_{i+1})} + k_i$$

Key-Recovery Attack on Jarvis III

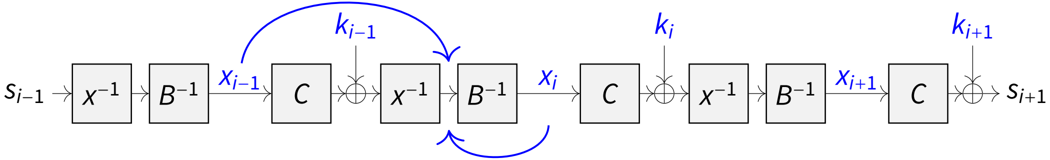


Basic equations

$$B(x_i) = \frac{1}{C(x_{i-1}) + k_{i-1}}$$

$$C(x_i) = \frac{1}{B(x_{i+1})} + k_i$$

Key-Recovery Attack on JARVIS III

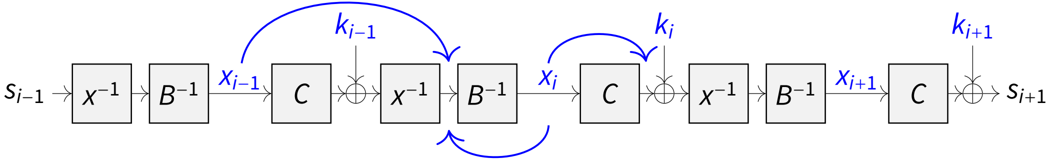


Basic equations

$$B(x_i) = \frac{1}{C(x_{i-1}) + k_{i-1}}$$

$$C(x_i) = \frac{1}{B(x_{i+1})} + k_i$$

Key-Recovery Attack on JARVIS III

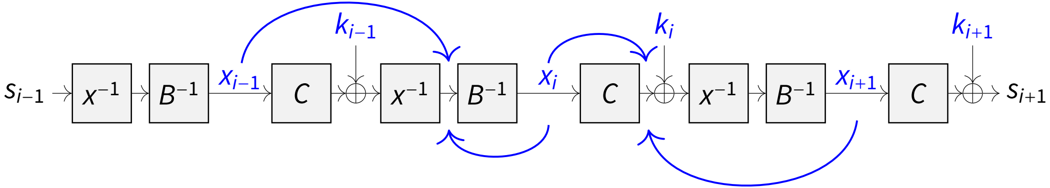


Basic equations

$$B(x_i) = \frac{1}{C(x_{i-1}) + k_{i-1}}$$

$$C(x_i) = \frac{1}{B(x_{i+1})} + k_i$$

Key-Recovery Attack on JARVIS III

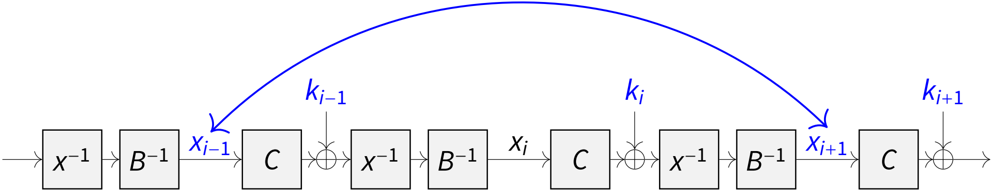


Basic equations

$$B(x_i) = \frac{1}{C(x_{i-1}) + k_{i-1}}$$

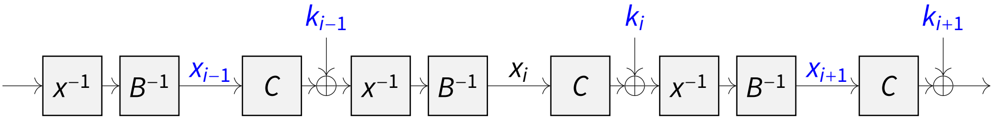
$$C(x_i) = \frac{1}{B(x_{i+1})} + k_i$$

Key-Recovery Attack on Jarvis IV



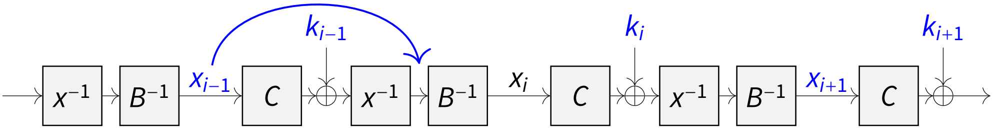
Idea for improvements: Only use every second intermediate state by finding affine polynomials B', C' such that $B' \circ B = C' \circ C$!

Key-Recovery Attack on JARVIS V



Improved equations

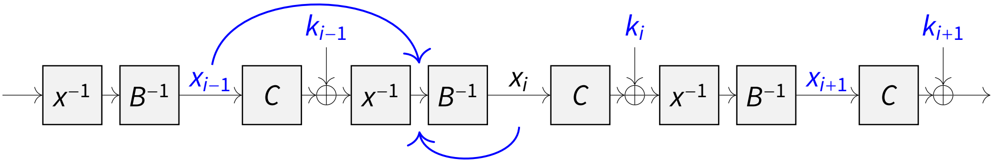
Key-Recovery Attack on JARVIS V



Improved equations

$$\frac{1}{C(x_{i-1}) + k_{i-1}} =$$

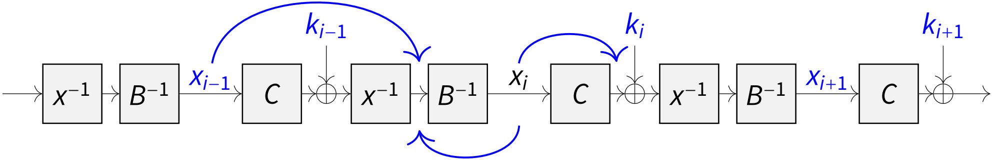
Key-Recovery Attack on JARVIS V



Improved equations

$$\frac{1}{C(x_{i-1}) + k_{i-1}} = B(x_i)$$

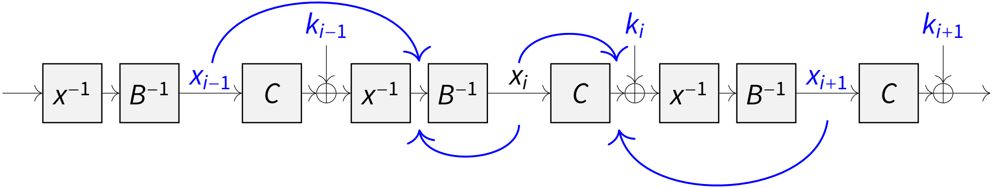
Key-Recovery Attack on JARVIS V



Improved equations

$$\frac{1}{C(x_{i-1}) + k_{i-1}} = B(x_i) \qquad C(x_i) =$$

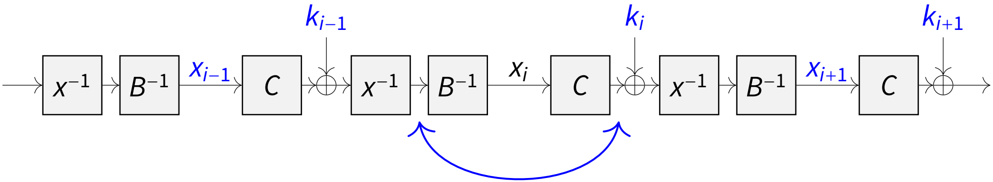
Key-Recovery Attack on JARVIS V



Improved equations

$$\frac{1}{C(x_{i-1}) + k_{i-1}} = B(x_i) \qquad C(x_i) = \frac{1}{B(x_{i+1})} + k_i$$

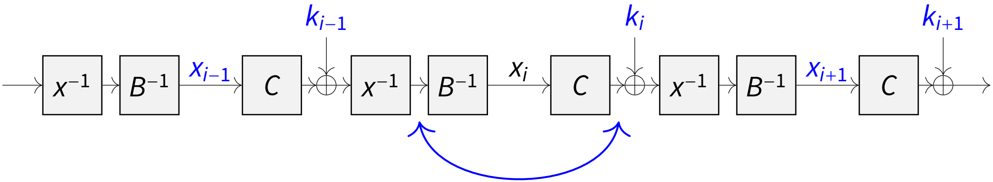
Key-Recovery Attack on Jarvis V



Improved equations

$$\frac{1}{C(x_{i-1}) + k_{i-1}} = B(x_i) \qquad C(x_i) = \frac{1}{B(x_{i+1})} + k_i$$

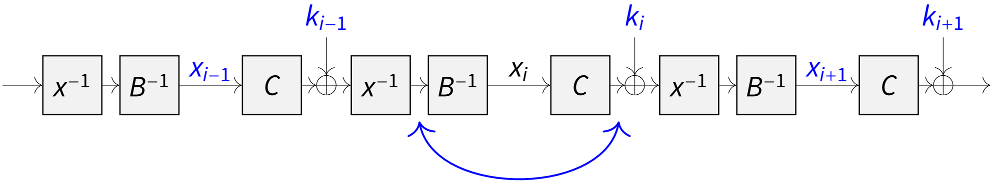
Key-Recovery Attack on Jarvis V



Improved equations

$$B'\left(\frac{1}{C(x_{i-1}) + k_{i-1}}\right) = B'(B(x_i)) \qquad C(x_i) = \frac{1}{B(x_{i+1})} + k_i$$

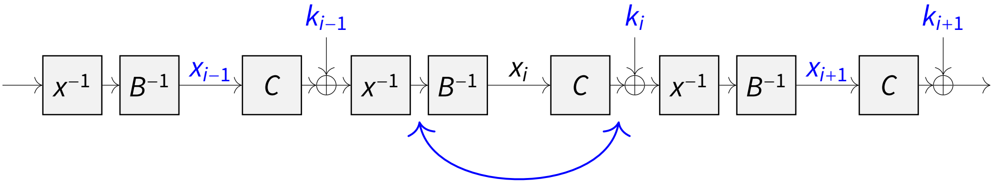
Key-Recovery Attack on Jarvis V



Improved equations

$$B'\left(\frac{1}{C(x_{i-1}) + k_{i-1}}\right) = B'(B(x_i)) \qquad C'(C(x_i)) = C'\left(\frac{1}{B(x_{i+1})} + k_i\right)$$

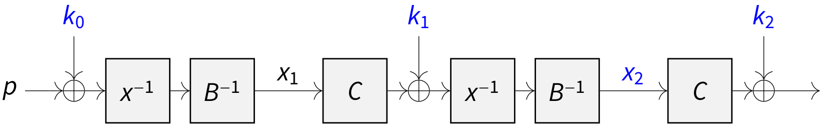
Key-Recovery Attack on JARVIS V



Improved equations

$$B'\left(\frac{1}{C(x_{i-1}) + k_{i-1}}\right) = B'(B(x_i)) \overset{!}{=} C'(C(x_i)) = C'\left(\frac{1}{B(x_{i+1})} + k_i\right)$$

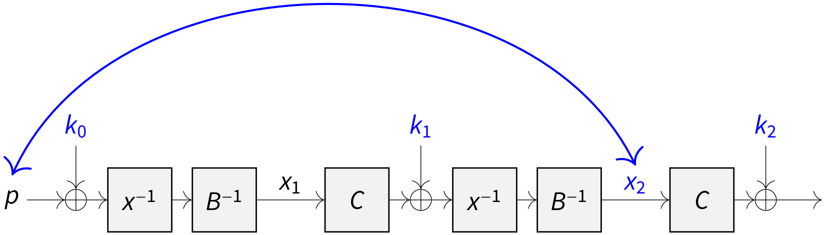
Relation to Plaintext



Plaintext equation

$$B'\left(\frac{1}{p+k_0}\right) = C'\left(\frac{1}{B(x_2)+k_1}\right)$$

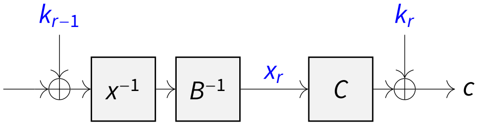
Relation to Plaintext



Plaintext equation

$$B'\left(\frac{1}{p + k_0}\right) = C'\left(\frac{1}{B(x_2) + k_1}\right)$$

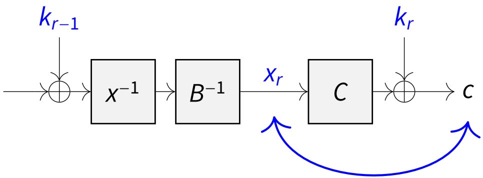
Relation to Ciphertext



Ciphertext equation

$$C(x_r) + k_r = c$$

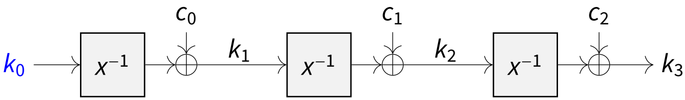
Relation to Ciphertext



Ciphertext equation

$$C(x_r) + k_r = c$$

Exploiting the Key Schedule



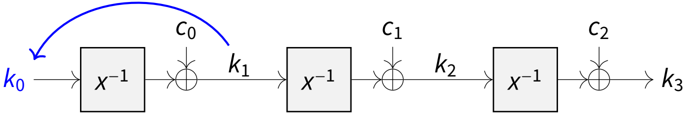
The first three round keys are given by

$$k_1 = \frac{1}{k_0} + c_0, \quad k_2 = \frac{1}{k_1} + c_1 = \frac{1}{\frac{1}{k_0} + c_0} + c_1, \quad k_3 = \frac{1}{k_2} + c_2 = \frac{1}{\frac{1}{\frac{1}{k_0}+c_0} + c_1};$$

more generally and after simplifying each fraction we have for $1 \leq i \leq r$

$$k_i = \frac{\alpha_i \cdot k_0 + \beta_i}{\gamma_i \cdot k_0 + \delta_i} \quad (\alpha_i, \beta_i, \gamma_i, \delta_i \in \mathbb{F}_{2^n}).$$

Exploiting the Key Schedule



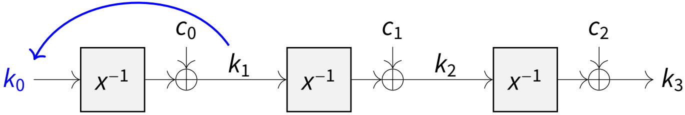
The first three round keys are given by

$$k_1 = \frac{1}{k_0} + c_0, \quad k_2 = \frac{1}{k_1} + c_1 = \frac{1}{\frac{1}{k_0} + c_0} + c_1, \quad k_3 = \frac{1}{k_2} + c_2 = \frac{1}{\frac{1}{\frac{1}{k_0}+c_0} + c_1};$$

more generally and after simplifying each fraction we have for $1 \le i \le r$

$$k_i = \frac{\alpha_i \cdot k_0 + \beta_i}{\gamma_i \cdot k_0 + \delta_i} \quad (\alpha_i, \beta_i, \gamma_i, \delta_i \in \mathbb{F}_{2^n}).$$

Exploiting the Key Schedule



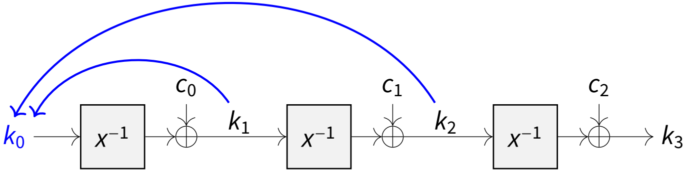
The first three round keys are given by

$$k_1 = \frac{1}{k_0} + c_0, \quad k_2 = \frac{1}{k_1} + c_1 = \frac{1}{\frac{1}{k_0} + c_0} + c_1, \quad k_3 = \frac{1}{k_2} + c_2 = \frac{1}{\frac{1}{\frac{1}{k_0}+c_0} + c_1};$$

more generally and after simplifying each fraction we have for $1 \leq i \leq r$

$$k_i = \frac{\alpha_i \cdot k_0 + \beta_i}{\gamma_i \cdot k_0 + \delta_i} \quad (\alpha_i, \beta_i, \gamma_i, \delta_i \in \mathbb{F}_{2^n}).$$

Exploiting the Key Schedule



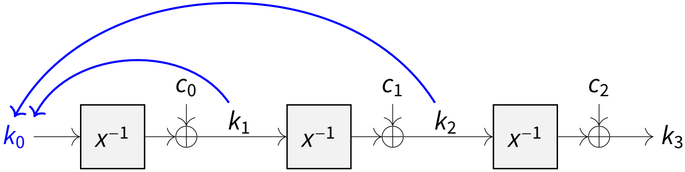
The first three round keys are given by

$$k_1 = \frac{1}{k_0} + c_0, \quad k_2 = \frac{1}{k_1} + c_1 = \frac{1}{\frac{1}{k_0} + c_0} + c_1, \quad k_3 = \frac{1}{k_2} + c_2 = \frac{1}{\frac{1}{\frac{1}{k_0}+c_0} + c_1};$$

more generally and after simplifying each fraction we have for $1 \le i \le r$

$$k_i = \frac{\alpha_i \cdot k_0 + \beta_i}{\gamma_i \cdot k_0 + \delta_i} \quad (\alpha_i, \beta_i, \gamma_i, \delta_i \in \mathbb{F}_{2^n}).$$

Exploiting the Key Schedule



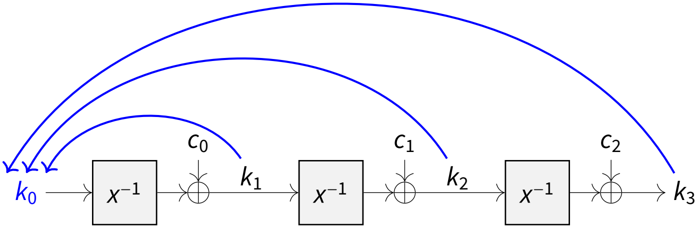
The first three round keys are given by

$$k_1 = \frac{1}{k_0} + c_0, \quad k_2 = \frac{1}{k_1} + c_1 = \frac{1}{\frac{1}{k_0} + c_0} + c_1, \quad k_3 = \frac{1}{k_2} + c_2 = \frac{1}{\frac{1}{\frac{1}{k_0}+c_0} + c_1};$$

more generally and after simplifying each fraction we have for $1 \leq i \leq r$

$$k_i = \frac{\alpha_i \cdot k_0 + \beta_i}{\gamma_i \cdot k_0 + \delta_i} \quad (\alpha_i, \beta_i, \gamma_i, \delta_i \in \mathbb{F}_{2^n}).$$

Exploiting the Key Schedule



The first three round keys are given by

$$k_1 = \frac{1}{k_0} + c_0, \quad k_2 = \frac{1}{k_1} + c_1 = \frac{1}{\frac{1}{k_0} + c_0} + c_1, \quad k_3 = \frac{1}{k_2} + c_2 = \frac{1}{\frac{1}{\frac{1}{k_0}+c_0} + c_1};$$

more generally and after simplifying each fraction we have for $1 \leq i \leq r$

$$k_i = \frac{\alpha_i \cdot k_0 + \beta_i}{\gamma_i \cdot k_0 + \delta_i} \quad (\alpha_i, \beta_i, \gamma_i, \delta_i \in \mathbb{F}_{2^n}).$$

Final Equation System for JARVIS

- Variables
  - $\frac{r}{2}$ variables for the intermediate states $x_2, x_4, \ldots, x_r$
  - 1 variable k_0 for the keys
- Equations
  - $\frac{r}{2} - 1$ equations for relating every second intermediate state
  - 2 equations for relating the plaintext p to x_2 and the ciphertext c to x_r

$\longrightarrow$ Solve this system with the help of Gröbner bases!

Final Equation System for JARVIS

- Variables
  - $\frac{r}{2}$ variables for the intermediate states $x_2, x_4, \ldots, x_r$
  - 1 variable k_0 for the keys
- Equations
  - $\frac{r}{2} - 1$ equations for relating every second intermediate state
  - 2 equations for relating the plaintext p to x_2 and the ciphertext c to x_r

$\longrightarrow$ Solve this system with the help of Gröbner bases!

Final Equation System for Jarvis

- Variables
  - $\frac{r}{2}$ variables for the intermediate states $x_2, x_4, \ldots, x_r$
  - 1 variable k_0 for the keys
- Equations
  - $\frac{r}{2} - 1$ equations for relating every second intermediate state
  - 2 equations for relating the plaintext p to x_2 and the ciphertext c to x_r

$\longrightarrow$ Solve this system with the help of Gröbner bases!

Final Equation System for JARVIS

- Variables
  - $\frac{r}{2}$ variables for the intermediate states $x_2, x_4, \ldots, x_r$
  - 1 variable k_0 for the keys
- Equations
  - $\frac{r}{2} - 1$ equations for relating every second intermediate state
  - 2 equations for relating the plaintext p to x_2 and the ciphertext c to x_r

$\longrightarrow$ Solve this system with the help of Gröbner bases!

Final Equation System for JARVIS

- Variables
  - $\frac{r}{2}$ variables for the intermediate states $x_2, x_4, \ldots, x_r$
  - 1 variable k_0 for the keys
- Equations
  - $\frac{r}{2} - 1$ equations for relating every second intermediate state
  - 2 equations for relating the plaintext p to x_2 and the ciphertext c to x_r

$\longrightarrow$ Solve this system with the help of Gröbner bases!

Attack complexity

Complexity estimates for Gröbner basis computation:

Rounds	Complexity Jarvis ($\log_2 \#ops$)	Complexity Friday ($\log_2 \#ops$)
6	45	34
8	58	47
10 (JARVIS-128)	**72**	**59**
12 (JARVIS-192)	**85**	**72**
14 (JARVIS-256)	**98**	**85**
16	112	97
18	125	110
20	138	123

Practical Results

Attack on Jarvis and Friday working over $\mathbb{F}_{2^{128}}$ implemented using Sage v8.6 with Magma v2.20-5 (using one core only).

Rounds	Jarvis Complex. ($\log_2 \#ops$)	Jarvis Time	Friday Complex. ($\log_2 \#ops$)	Friday Time
3	20	0.3 s	19	3.6 s
4	31	9.4 s	22	0.5 s
5	34	14.9 min	32	36.5 s
6	45	27.8 h	34	34.9 min

Most of the time, our attacks performed substantially better in practice than the complexity estimates suggest.

Practical Results

Attack on Jarvis and Friday working over $\mathbb{F}_{2^{128}}$ implemented using Sage v8.6 with Magma v2.20-5 (using one core only).

Rounds	Jarvis Complex. ($\log_2 \#ops$)	Jarvis Time	Friday Complex. ($\log_2 \#ops$)	Friday Time
3	20	0.3 s	19	3.6 s
4	31	9.4 s	22	0.5 s
5	34	14.9 min	32	36.5 s
6	45	27.8 h	34	34.9 min

Most of the time, our attacks performed substantially better in practice than the complexity estimates suggest.

Conclusion

The main reason why MARVELlous is **less secure** than claimed is

- the particular usage of two **low-degree polynomials** as affine layer,
- together with **finite field inversion** as non-linear layer.

MiMC is **immune** against the presented attack strategy because

- factoring the univariate polynomial is prohibitively expensive;
- although the polynomials representing MiMC are already a Gröbner basis.

Conclusion

The main reason why MARVELlous is **less secure** than claimed is

- the particular usage of two **low-degree polynomials** as affine layer,
- together with **finite field inversion** as non-linear layer.

MiMC is **immune** against the presented attack strategy because

- factoring the univariate polynomial is prohibitively expensive;
- although the polynomials representing MiMC are already a Gröbner basis.

Outlook

Other Designs: GMiMC [AGP+19], **Starkad&Poseidon** [GKK+19] (based on **Hades** [GLR+19]), **Vision&Rescue** [AABS+19]

Ongoing Competition: STARK-friendly Hash-Challenge

https://starkware.co/hash-challenge/

Questions?

References I

[AABS+19] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, et al. **Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols.** Cryptology ePrint Archive, Report 2019/426. https://eprint.iacr.org/2019/426. 2019 (cit. on p. 58).

[AD18] Tomer Ashur and Siemen Dhooghe. **MARVELlous: a STARK-Friendly Family of Cryptographic Primitives.** Cryptology ePrint Archive, Report 2018/1098. https://eprint.iacr.org/2018/1098. 2018 (cit. on pp. 11–14).

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, et al. **Feistel Structures for MPC, and More.** ESORICS 2019: 24th European Symposium on Research in Computer Security. https://eprint.iacr.org/2019/397. 2019 (cit. on p. 58).

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, et al. **MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity.** ASIACRYPT 2016, Part I. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. Springer, Heidelberg, Dec. 2016, pp. 191–219. DOI: 10.1007/978-3-662-53887-6_7 (cit. on p. 17).

References II

[BBH+18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, et al. **Scalable, transparent, and post-quantum secure computational integrity.** IACR Cryptology ePrint Archive 2018 (2018), p. 46 (cit. on pp. 15, 16).

[GKK+19] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, et al. **Starkad and Poseidon: New Hash Functions for Zero Knowledge Proof Systems.** Cryptology ePrint Archive, Report 2019/458. https://eprint.iacr.org/2019/458. 2019 (cit. on p. 58).

[GLR+19] Lorenzo Grassi, Reinhard Lueftenegger, Christian Rechberger, et al. **On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy.** Cryptology ePrint Archive, Report 2019/1107. https://eprint.iacr.org/2019/1107. 2019 (cit. on p. 58).