Implementing Joux-Vitse's Crossbred Algorithm for Solving MQ Systems over $\mathbb{F}_2$ on GPUs

Ruben Niederhagen[1], Kai-Chun Ning[2], Bo-Yin Yang[3]

1. Fraunhofer SIT
2. Technische Universiteit Eindhoven
3. Academia Sinica

9 April, 2018

- Applications: post-quantum cryptography

- Applications: post-quantum cryptography
  - Encryption: HFE, ZHFE, Square

Technische Universiteit Eindhoven University of Technology

- Applications: post-quantum cryptography
  - Encryption: HFE, ZHFE, Square
  - Signature: SOFIA, SFLASH, UOV, QUARTZ, Rainbow

MP and MQ Systems

- Applications: post-quantum cryptography
  - Encryption: HFE, ZHFE, Square
  - Signature: SOFIA, SFLASH, UOV, QUARTZ, Rainbow
  - Stream cipher: QUAD

MP and MQ Systems

- Applications: post-quantum cryptography
  - Encryption: HFE, ZHFE, Square
  - Signature: SOFIA, SFLASH, UOV, QUARTZ, Rainbow
  - Stream cipher: QUAD
- Cryptanalysis on MQ cryptographic schemes

MP and MQ Systems

- Applications: post-quantum cryptography
  - Encryption: HFE, ZHFE, Square
  - Signature: SOFIA, SFLASH, UOV, QUARTZ, Rainbow
  - Stream cipher: QUAD
- Cryptanalysis on MQ cryptographic schemes
  - Solve MQ systems that are not completely random

MP and MQ Systems

- Applications: post-quantum cryptography
  - Encryption: HFE, ZHFE, Square
  - Signature: SOFIA, SFLASH, UOV, QUARTZ, Rainbow
  - Stream cipher: QUAD
- Cryptanalysis on MQ cryptographic schemes
  - Solve MQ systems that are not completely random
  - Scheme-specific

Goal

Generic attack on MQ over $\mathbb{F}_2$

- Schemes built on MQ over $\mathbb{F}_2$: UOV, MQ-hash, MQ-PKI

Goal

Generic attack on MQ over $\mathbb{F}_2$

- Schemes built on MQ over $\mathbb{F}_2$: UOV, MQ-hash, MQ-PKI
- Substitute variables and introducing equations: MP $\rightarrow$ MQ

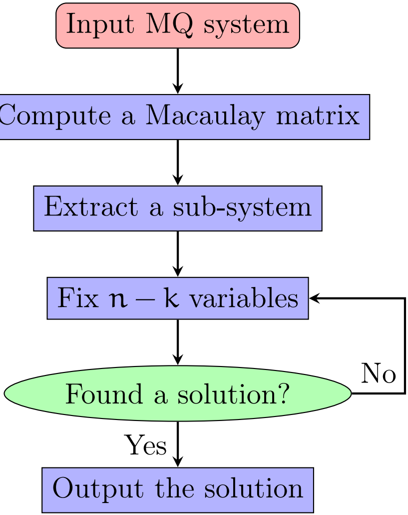
Goal

Generic attack on MQ over $\mathbb{F}_2$

- Schemes built on MQ over $\mathbb{F}_2$: UOV, MQ-hash, MQ-PKI
- Substitute variables and introducing equations: MP $\rightarrow$ MQ
- With Weil descent: MQ over $\mathbb{F}_{2^n} \rightarrow$ MQ over $\mathbb{F}_2$

Technische Universiteit Eindhoven University of Technology

The Original Crossbred Algorithm



Crossbred - Macaulay Matrix

- Definition

- Definition
  - A Macaulay matrix: a system of polynomials extended from a base system $\mathcal{F}$ of m polynomials of degree d in n variables.

Technische Universiteit Eindhoven University of Technology

- Definition
  - A Macaulay matrix: a system of polynomials extended from a base system $\mathcal{F}$ of m polynomials of degree d in n variables.
  - The Macaulay degree D: the maximal degree of the polynomials in the extended system.

Technische Universiteit Eindhoven University of Technology

Crossbred - Macaulay Matrix

- Definition
  - A Macaulay matrix: a system of polynomials extended from a base system $\mathcal{F}$ of m polynomials of degree d in n variables.
  - The Macaulay degree D: the maximal degree of the polynomials in the extended system.
  - Each row in a Macaulay matrix is the product of a polynomial f in $\mathcal{F}$ by a monomial t such that $\deg(t \cdot f) \leqslant D$.

Technische Universiteit Eindhoven University of Technology

- Definition
  - A Macaulay matrix: a system of polynomials extended from a base system $\mathcal{F}$ of m polynomials of degree d in n variables.
  - The Macaulay degree D: the maximal degree of the polynomials in the extended system.
  - Each row in a Macaulay matrix is the product of a polynomial f in $\mathcal{F}$ by a monomial t such that $\deg(t \cdot f) \leqslant D$.
- Example

$$\mathcal{F} = \begin{cases} f_1 = x_1x_2 + x_2x_3 + x_3 \\ f_2 = x_1x_4 + 1 \end{cases}$$

Technische Universiteit Eindhoven University of Technology

Crossbred - Macaulay Matrix

- Row order: graded reverse lexicographical order w.r.t to the multipliers

Crossbred - Macaulay Matrix

- Row order: graded reverse lexicographical order w.r.t to the multipliers
- Column order: graded reverse lexicographical order

Crossbred - Macaulay Matrix

- Row order: graded reverse lexicographical order w.r.t to the multipliers
- Column order: graded reverse lexicographical order

	$x_1x_2x_3x_4$	$x_1x_2x_3$	$x_1x_2x_4$	$x_1x_3x_4$	$x_2x_3x_4$	x_1x_2	x_1x_3	x_2x_3	x_1x_4	x_2x_4	x_3x_4	x_1	x_2	x_3	x_4	1
$x_1x_2 \cdot f_1$						1										
$x_1x_2 \cdot f_2$			1			1										
$x_1x_3 \cdot f_1$							1									
$x_1x_3 \cdot f_2$				1			1									
$x_2x_3 \cdot f_1$		1														
$x_2x_3 \cdot f_2$	1							1								
$x_1x_4 \cdot f_1$	1		1	1												
$x_1x_4 \cdot f_2$																
$x_2x_4 \cdot f_1$			1													
$x_2x_4 \cdot f_2$			1							1						
$x_3x_4 \cdot f_1$	1				1						1					
$x_3x_4 \cdot f_2$				1							1					
$x_1 \cdot f_1$		1				1	1									
$x_1 \cdot f_2$									1			1				
$x_2 \cdot f_1$						1										
$x_2 \cdot f_2$			1										1			
$x_3 \cdot f_1$		1						1						1		
$x_3 \cdot f_2$				1										1		
$x_4 \cdot f_1$			1		1						1					
$x_4 \cdot f_2$									1						1	
$1 \cdot f_1$						1		1						1		
$1 \cdot f_2$									1							1

-

Crossbred - Extracted Sub-system

- Consists of only monomials that become linear in $x_1, \ldots x_k$ after fixing $x_{k+1}, \ldots, x_n$.

Crossbred - Extracted Sub-system

- Consists of only monomials that become linear in $x_1, \ldots x_k$ after fixing $x_{k+1}, \ldots, x_n$.
- Example: Fixing $(x_3, x_4) = (0, 0)$ in

$$\mathcal{S} = \begin{cases} x_1x_4 + x_2x_3 + x_1 + x_3 + x_4 = 0 \\ x_1x_3 + x_3x_4 + x_2 + 1 = 0 \\ x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_4 = 0 \end{cases}$$

yields

$$\mathcal{S}' = \begin{cases} x_1 = 0 \\ x_2 + 1 = 0 \\ x_1 = 0 \end{cases}$$

Technische Universiteit Eindhoven University of Technology

- Basic idea:
  $x_1x_2 + x_1x_3 + x_2x_3 + x_3 + 1 = (x_1x_2 + 1) + x_3(x_1 + x_2 + 1)$

Crossbred - Fast Evaluate

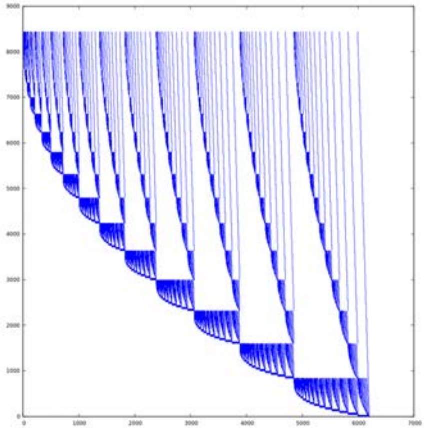
- Basic idea:
  $x_1x_2 + x_1x_3 + x_2x_3 + x_3 + 1 = (x_1x_2 + 1) + x_3(x_1 + x_2 + 1)$
- Apply this technique recursively to fix $n - k$ variables

- How to extract a sub-system?

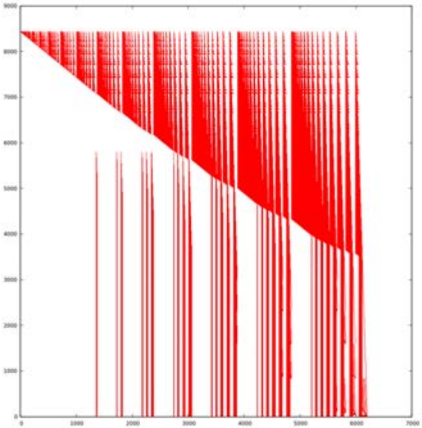
- How to extract a sub-system?
- How to check the solvability of a linear system?

Technische Universiteit Eindhoven University of Technology

Gaussian elimination?



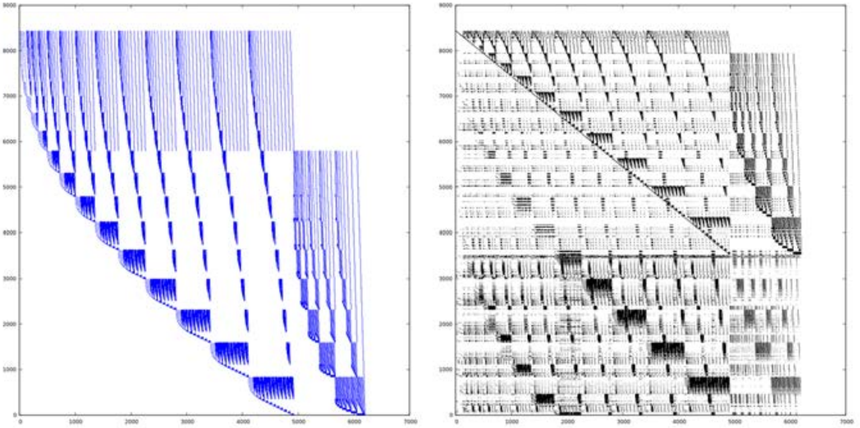
(a) Original Macaulay Matrix



(b) After Gaussian Elimination

Technische Universiteit Eindhoven University of Technology

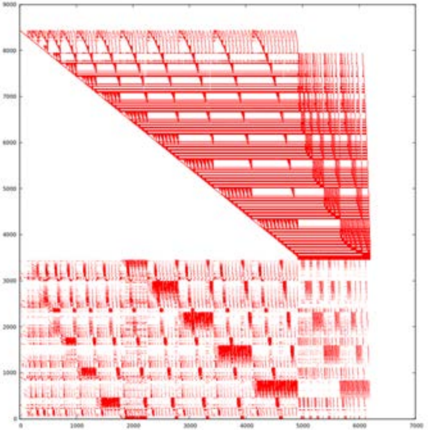
Permute columns and swap rows



(a) Permuted Macaulay Matrix

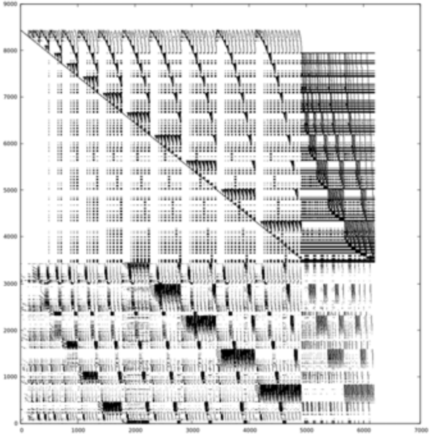
(b) After Row-swapping

Ignore the lower part



(c) After Gaussian Elimination

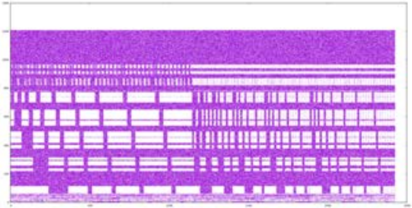
Reduce dimension



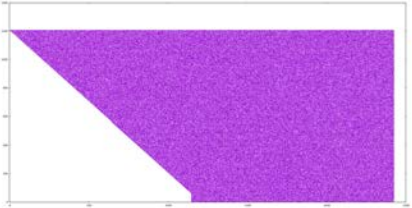
(d) After Eliminating Pivot Monomials

Technische Universiteit Eindhoven University of Technology

- Reduced Macaulay matrix



(a) Initial Reduced Macaulay Matrix

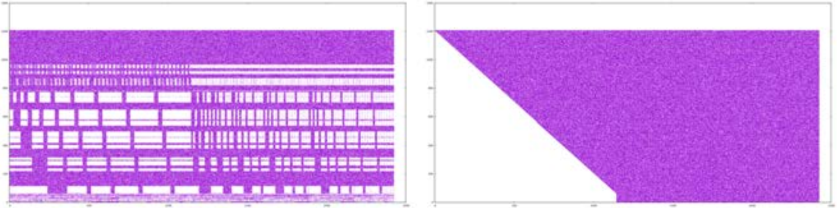


(b) After Gaussian Elimination

Technische Universiteit Eindhoven University of Technology

Adaption - Extracting a Sub-system

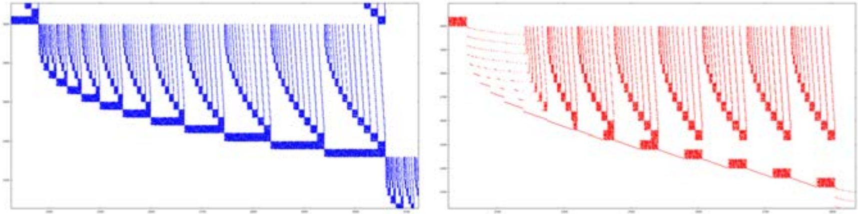
- Reduced Macaulay matrix



(a) Initial Reduced Macaulay Matrix

(b) After Gaussian Elimination

- A rule of thumb: Gauss-Jordan elimination



(a) Without Gauss-Jordan Elimination

(b) With Gauss-Jordan Elimination

- Observation:

$$f(\vec{a}) = f(\vec{a'}) + \frac{\partial f}{\partial x_i}(\vec{a'}).$$

when $\vec{a}$ and $\vec{a'}$ differs only by one coordinate.

Adaption - Fixing Variables in the Sub-system

- Observation:

$$f(\vec{a}) = f(\vec{a'}) + \frac{\partial f}{\partial x_i}(\vec{a'}).$$

  when $\vec{a}$ and $\vec{a'}$ differs only by one coordinate.
- For sub-system $\mathcal{S}$ of m degree-D equations in n variables, fix $n - k$ variables in $\mathcal{S}$ takes $\mathcal{O}(D \cdot k)$ instructions.

Example

$$f(x_4, x_5, x_6, x_7) = x_1x_4x_5x_6 + x_1x_4x_5x_7 + x_4x_5x_6x_7 + x_1x_4x_5 + x_2x_4x_6 + x_4x_6x_7 + x_1x_4 + x_1x_5 + x_5x_7 + x_6x_7 + x_1 + x_2 + x_4 + 1$$

$$\frac{\partial f}{\partial x_4} = x_1x_5x_6 + x_1x_5x_7 + x_5x_6x_7 + x_1x_5 + x_2x_6 + x_6x_7 + x_1 + 1$$

$$\frac{\partial^2 f}{\partial x_4 \partial x_7} = x_1x_5 + x_5x_6 + x_6$$

$$\frac{\partial^3 f}{\partial x_4 \partial x_6 \partial x_7} = x_5 + 1$$

Adaption - Fixing Variables in the Sub-system

$$
\begin{aligned}
f(0,0,0,0) &= f(1,0,0,0) + \frac{\partial f}{\partial x_4}(1,0,0,0) \\
&= f(1,0,0,0) + \frac{\partial f}{\partial x_4}(1,0,0,1) + \frac{\partial^2 f}{\partial x_4 \partial x_7}(1,0,0,1) \\
&= f(1,0,0,0) + \frac{\partial f}{\partial x_4}(1,0,0,1) + \frac{\partial^2 f}{\partial x_4 \partial x_7}(1,0,1,1) + \frac{\partial^3 f}{\partial x_4 \partial x_6 \partial x_7}(1,0,1,1) \\
&= f(1,0,0,0) + \frac{\partial f}{\partial x_4}(1,0,0,1) + \frac{\partial^2 f}{\partial x_4 \partial x_7}(1,0,1,1) + \frac{\partial^3 f}{\partial x_4 \partial x_6 \partial x_7}(1,0,1,0) + \frac{\partial^4 f}{\partial x_4 \partial x_6 \partial x_7 \partial x_7} \\
&= f(1,0,0,0) + \frac{\partial f}{\partial x_4}(1,0,0,1) + \frac{\partial^2 f}{\partial x_4 \partial x_7}(1,0,1,1) + \frac{\partial^3 f}{\partial x_4 \partial x_6 \partial x_7}(1,0,1,0) + 0 \\
&= f(1,0,0,0) + \frac{\partial f}{\partial x_4}(1,0,0,1) + \frac{\partial^2 f}{\partial x_4 \partial x_7}(1,0,1,1) + 1 \\
&= f(1,0,0,0) + \frac{\partial f}{\partial x_4}(1,0,0,1) + 1 + 1 \\
&= f(1,0,0,0) + x_1 + 1 + 1 + 1 \\
&= x_2 + x_1 + 1 + 1 + 1 \\
&= x_1 + x_2 + 1
\end{aligned}
$$

because $1010 \to 1011 \to 1001 \to 1000 \to 0000$

TU/e Technische Universiteit Eindhoven University of Technology

Adaption - Testing a Linear System

Gauss-Jordan elimination?

$$\mathcal{S} = \begin{array}{c} \\ eq_1 \\ eq_2 \\ eq_3 \\ eq_4 \\ eq_5 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & 1 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array} \xrightarrow[\text{no swapping}]{1^{\text{st}} \text{ pivot element}} \begin{array}{c} eq_1 \\ eq_2 \\ eq_3 \\ eq_4 \\ eq_5 \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \xrightarrow[eq_3,\ eq_4]{\text{reduce with}} \begin{array}{c} eq_1 \\ eq_2 \\ eq_3 \\ eq_4 \\ eq_5 \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$\xrightarrow[\text{swap } eq_2 \text{ and } eq_3]{2^{\text{nd}} \text{ pivot element}} \begin{array}{c} eq_1 \\ eq_3 \\ eq_2 \\ eq_4 \\ eq_5 \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \xrightarrow[eq_1,\ eq_4,\ eq_5]{\text{reduce with}} \begin{array}{c} eq_1 \\ eq_3 \\ eq_2 \\ eq_4 \\ eq_5 \end{array} \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\xrightarrow[\text{swap } eq_2 \text{ and } eq_4]{3^{\text{rd}} \text{ pivot element}} \begin{array}{c} eq_1 \\ eq_3 \\ eq_4 \\ eq_2 \\ eq_5 \end{array} \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \xrightarrow[eq_3,\ eq_5]{\text{reduce with}} \begin{array}{c} eq_1 \\ eq_3 \\ eq_4 \\ eq_2 \\ eq_5 \end{array} \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\xrightarrow[\text{no swapping}]{4^{\text{th}} \text{ pivot element}} \begin{array}{c} eq_1 \\ eq_3 \\ eq_4 \\ eq_2 \\ eq_5 \end{array} \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \xrightarrow[eq_1,\ eq_5]{\text{reduce with}} \begin{array}{c} eq_1 \\ eq_3 \\ eq_4 \\ eq_2 \\ eq_5 \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A better variant:

- $\mathcal{O}(k^2)$ instructions for testing
- $\mathcal{O}(k)$ instructions for extracting a solution

$$
\begin{array}{c} \text{mask} \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{array}
\quad
\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}
\xrightarrow[\text{reduce with } eq_3,\ eq_4]{1^{\text{st}} \text{ pivot row: } eq_1}
\begin{array}{c} 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{array}
\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}
\xrightarrow[\text{reduce with } eq_1,\ eq_4,\ eq_5]{2^{\text{nd}} \text{ pivot row: } eq_3}
\begin{array}{c} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{array}
\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}
$$

$$
\xrightarrow[\text{reduce with } eq_3,\ eq_5]{3^{\text{rd}} \text{ pivot row: } eq_4}
\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{array}
\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}
\xrightarrow[\text{reduce with } eq_1,\ eq_5]{4^{\text{th}} \text{ pivot row: } eq_2}
\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{array}
\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}
$$

- Split search space into N smaller sub-spaces and launch N threads?

Adaption - Parallelization

- Split search space into N smaller sub-spaces and launch N threads?
  - Different starting points $\rightarrow$ not ideal for GPU because divergent paths

Adaption - Parallelization

- Split search space into N smaller sub-spaces and launch N threads?
  - Different starting points → not ideal for GPU because divergent paths
- Fix t variables in the sub-system to create 2^t smaller sub-systems

Technische Universiteit Eindhoven University of Technology

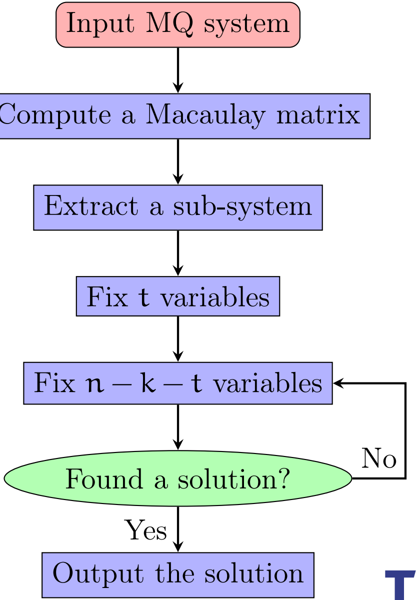
- Split search space into N smaller sub-spaces and launch N threads?
  - Different starting points → not ideal for GPU because divergent paths
- Fix t variables in the sub-system to create 2^t smaller sub-systems
  - Same enumeration starting point

Adaption - Parallelization

- Split search space into N smaller sub-spaces and launch N threads?
  - Different starting points → not ideal for GPU because divergent paths
- Fix t variables in the sub-system to create 2^t smaller sub-systems
  - Same enumeration starting point
  - Same last order partial derivatives

TU/e Technische Universiteit Eindhoven University of Technology

The Adapted Crossbred Algorithm



- From scratch

MQsolver

- From scratch
- Pure C

MQsolver

- From scratch
- Pure C
- Generate C code with Python script

- From scratch
- Pure C
- Generate C code with Python script
- Part on CPU, part on GPU $\rightarrow$ pipeline

Environment

The Saber cluster:

https://blog.cr.yp.to/20140602-saber.html

- GTX 780 / GTX 980
- AMD FX-8350 @4GHz

Technische Universiteit
Eindhoven
University of Technology

Fukuoka MQ type 1 challenges: $n = 55$ to 74, $m = 2n$

Technische Universiteit Eindhoven University of Technology

$m = 2n$

For $n = 67$:

- Crossbred: 6200 CPU-hours (Xeon 2690)

For $n = 74$:

Technische Universiteit Eindhoven University of Technology

$m = 2n$

For $n = 67$:

- Crossbred: 6200 CPU-hours (Xeon 2690)
- Parallel Crossbred: 98.39 GPU-hours (GTX 980)

For $n = 74$:

$m = 2n$

For $n = 67$:

- Crossbred: 6200 CPU-hours (Xeon 2690)
- Parallel Crossbred: 98.39 GPU-hours (GTX 980)

For $n = 74$:

- Crossbred: $360,000$ CPU-hours

Technische Universiteit Eindhoven University of Technology

$m = 2n$

For $n = 67$:

- Crossbred: 6200 CPU-hours (Xeon 2690)
- Parallel Crossbred: 98.39 GPU-hours (GTX 980)

For $n = 74$:

- Crossbred: $360,000$ CPU-hours
- Parallel Crossbred: 8236.05 GPU-hours

- "Public Key Identification Schemes based on Multivariate Quadratic Polynomials", Koichi Sakumoto, Taizo Shirai, Harunaga Hiwatari, CRYPTO 2011

- "Public Key Identification Schemes based on Multivariate Quadratic Polynomials", Koichi Sakumoto, Taizo Shirai, Harunaga Hiwatari, CRYPTO 2011
- MQ system where $n = 84$, $m = 80$

Breaking a Post-quantum Cryptographic Scheme

- "Public Key Identification Schemes based on Multivariate Quadratic Polynomials", Koichi Sakumoto, Taizo Shirai, Harunaga Hiwatari, CRYPTO 2011
- MQ system where $n = 84$, $m = 80$
- GPU: Nvidia GTX 980

Technische Universiteit Eindhoven University of Technology

Breaking a Post-quantum Cryptographic Scheme

- "Public Key Identification Schemes based on Multivariate Quadratic Polynomials", Koichi Sakumoto, Taizo Shirai, Harunaga Hiwatari, CRYPTO 2011
- MQ system where $n = 84$, $m = 80$
- GPU: Nvidia GTX 980
- At most 37000, on average: 3600 (GPU-years)

Breaking a Post-quantum Cryptographic Scheme

- "Public Key Identification Schemes based on Multivariate Quadratic Polynomials", Koichi Sakumoto, Taizo Shirai, Harunaga Hiwatari, CRYPTO 2011
- MQ system where $n = 84$, $m = 80$
- GPU: Nvidia GTX 980
- At most 37000, on average: 3600 (GPU-years)
- Claimed to have 80 bits of security, in fact only 76.5 bits

Conslusion

- Parallel Crossbred outperforms the orginal Crossbred algorithm by a factor as high as ≈ 60.

Conslusion

- Parallel Crossbred outperforms the orginal Crossbred algorithm by a factor as high as ≈ 60.
- We solved all the Fukuoka MQ type I challenges, $n = 55$ to 74.

Conslusion

- Parallel Crossbred outperforms the orginal Crossbred algorithm by a factor as high as ≈ 60.
- We solved all the Fukuoka MQ type I challenges, $n = 55$ to 74.
- We showed that a post-quantum cryptographic scheme whose security relies on an MQ system where $n = 84$ and $m = 80$ can be broken with 37000 GPUs in at most one year, and on average 35 days.

TU/e Technische Universiteit Eindhoven University of Technology

Conslusion

- Parallel Crossbred outperforms the orginal Crossbred algorithm by a factor as high as ≈ 60.
- We solved all the Fukuoka MQ type I challenges, $n = 55$ to 74.
- We showed that a post-quantum cryptographic scheme whose security relies on an MQ system where $n = 84$ and $m = 80$ can be broken with 37000 GPUs in at most one year, and on average 35 days.
- We showed that an MQ system with only 80 bits of security should not be considered secure anymore.

Thank you for attention!

Technische Universiteit Eindhoven University of Technology