Software Toolkit for HFE-based Multivariate Schemes

J-C. Faugère[1,2], L. Perret[1,2], Jocelyn Ryckeghem[2]

[1]CryptoNext Security
[2]Sorbonne Université, CNRS, INRIA, LIP6, Équipe PolSys, F-75005 Paris, France

CHES, Atlanta, August 26, 2019

MQsoft[1]: Multivariate Quadratic Software

Motivations

- 11/2017 and 01/2019: beginning of the 1^{st} and 2^{nd} rounds of the NIST post-quantum cryptography standardization process.
- Signature: 4 second round candidates over 9 are multivariate.
- Libraries: code [McBits, CHES'2013, ...], lattice [NFLlib, CT RSA'16, ...], but no library for the multivariate-based schemes!

[1] https://www-polsys.lip6.fr/Links/NIST/MQsoft.html

MQsoft[1]: Multivariate Quadratic Software

Motivations

- 11/2017 and 01/2019: beginning of the 1st and 2nd rounds of the NIST post-quantum cryptography standardization process.
- Signature: 4 second round candidates over 9 are multivariate.
- Libraries: code [McBits, CHES'2013, ...], lattice [NFLlib, CT RSA'16, ...], but no library for the multivariate-based schemes!

Our contribution: MQsoft

- An efficient C library exploiting SSE and AVX2 instructions set.
- Matsumoto-Imai-based schemes: QUARTZ, Gui, GeMSS.
- Fast arithmetic in $\mathbb{F}_2[X]$, $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^n}[X]$ (with root finding), multivariate quadratic systems in $\mathbb{F}_2$ (evaluation, change of variables, ...), constant-time implementation against timing attacks (as often as possible).

[1] https://www-polsys.lip6.fr/Links/NIST/MQsoft.html

Matsumoto-Imai-based schemes

Matsumoto-Imai [EUROCRYPT '88]

- Public-key: a multivariate quadratic system.
- Example in $\mathbb{F}_2$: $\mathbf{p}(x_1, x_2, x_3) = \begin{cases} x_1x_2 + x_2x_3 + x_1 + 1 \\ x_1x_2 + x_1x_3 + x_1 \end{cases}$
- Verifying process: evaluation of the public-key.
- Signing process: affine transformations + inversion of the private map.

Matsumoto-Imai-based schemes

Matsumoto-Imai [EUROCRYPT '88]

- Public-key: a multivariate quadratic system.
- Example in $\mathbb{F}_2$: $\mathbf{p}(x_1, x_2, x_3) = \begin{cases} x_1x_2 + x_2x_3 + x_1 + 1 \\ x_1x_2 + x_1x_3 + x_1 \end{cases}$
- Verifying process: evaluation of the public-key.
- Signing process: affine transformations + inversion of the private map.

HFE-based signature schemes [Patarin, EUROCRYPT '96]

- Signing process: to find the roots of a univariate polynomial.
- Schemes: QUARTZ (2001), Gui (2015), GeMSS (2017), DualModeMS (2017), BlueGeMSS (2019), RedGeMSS (2019).

Performance

QUARTZ (a NESSIE submission)

- In 2001: $4s$ to generate the keys, $10s$ to sign, $900\mu s$ to verify.
- With MQsoft (new hardware + new library): $2.0ms$ to generate the keys, $20ms$ to sign, $6.4\mu s$ to verify.

Performance

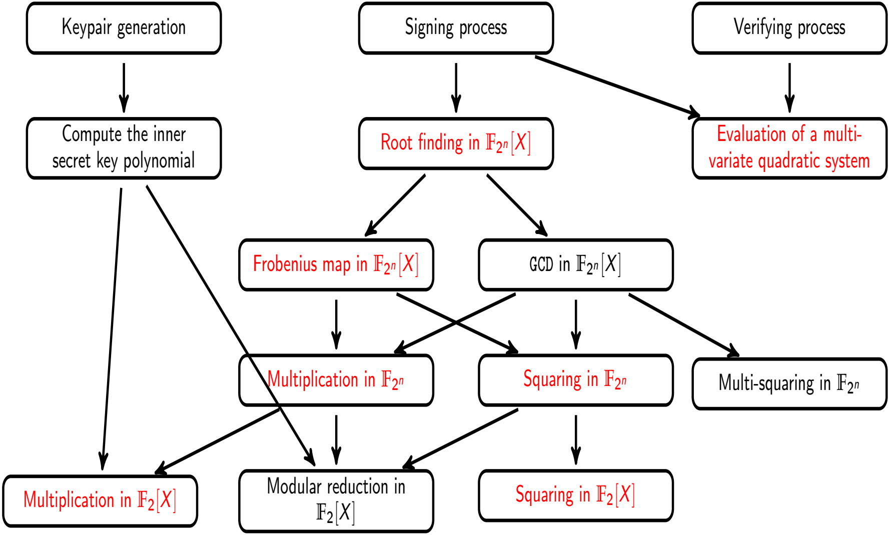
QUARTZ (a NESSIE submission)

- In 2001: 4s to generate the keys, 10s to sign, 900μs to verify.
- With MQsoft (new hardware + new library): 2.0ms to generate the keys, 20ms to sign, 6.4μs to verify.

sign. scheme	sec. level	key gen.	sign.	verif.
GeMSS128	128	+220%	+100%	+95%
GeMSS192	192	+220%	+57%	+84%
GeMSS256	256	+240%	+110%	+75%
Gui-184	128	+1200%	+100%	+73%
Gui-312	192	+1600%	+95%	+56%
Gui-448	256	+2500%	+85%	+58%

Speed-up (best first round implementations compared to MQsoft), Haswell processor. Speed-up of 100% for the signing process, and between 60% and 100% for the verifying process.

MQsoft: architecture for HFE



Efficient arithmetic in $\mathbb{F}_{2^n}$

Software and libraries for number theory

- `Magma`, a computer algebra software.
- `NTL`, A Library for Doing Number Theory (in `C++`).
- `FLINT`, Fast Library for Number Theory, **less efficient in $\mathbb{F}_{2^n}$!**
- `gf2x` (`C` library), specialized for the multiplication in $\mathbb{F}_2[X]$.

Implementations for specific fields

- Elliptic curves [BluGue13]: $\mathbb{F}_{2^{163}}$, $\mathbb{F}_{2^{233}}$, $\mathbb{F}_{2^{283}}$, ...
- `Gui` [mpkc-128bit, gui-pq-submission]: $\mathbb{F}_{2^{184}}$, $\mathbb{F}_{2^{240}}$, $\mathbb{F}_{2^{312}}$, ...

`MQsoft`

- Arithmetic in $\mathbb{F}_{2^n}$ for $n \leq 576$, in `C` using `AVX2` instructions set.
- Especially efficient on Skylake processors (6th generation), but also efficient on Haswell processors (4th generation).

Constant-time product in $\mathbb{F}_{2^n} = \mathbb{F}_2[X]/f(x)$

Code using SSE (128 bits) or AVX2 (256 bits) instructions sets.

Multiplication

The most important operation!

1. School-book algorithm by block of 64 bits (PCLMULQDQ).
2. Karatsuba algorithm, the base case depends on the processor.

n	Magma	NTL	MQsoft
252	558	169	36-40
511	761	320	91-92

Multiplication in $\mathbb{F}_{2^n}$ in cycles, Skylake processor.

Squaring

Linear operation in char. 2: $(ax+b)^2 = a^2x^2 + b^2$.

1. Table lookups of square (PSHUFB, VPSHUFB).
2. Squaring of each 64-bit block (PCLMULQDQ).

n	Magma	NTL	MQsoft
252	455	128	15-24
511	510	174	24-27

Squaring in $\mathbb{F}_{2^n}$ in cycles, Skylake processor.

Representation of multivariate quadratic systems (m equations, n variables)

Representation "equation by equation"

- The equations are stored one by one.
- Example in $\mathbb{F}_2$: $\mathbf{p}(x_1, x_2, x_3) = \begin{cases} x_1x_2 + x_2x_3 + x_1 + 1 & (1) \\ x_1x_2 + x_1x_3 + x_1 & (2) \end{cases}$

Representation of multivariate quadratic systems (m equations, n variables)

Representation "equation by equation"

- The equations are stored one by one.
- Example in $\mathbb{F}_2$: $\mathbf{p}(x_1, x_2, x_3) = \begin{cases} x_1x_2 + x_2x_3 + x_1 + 1 & (1) \\ x_1x_2 + x_1x_3 + x_1 & (2) \end{cases}$

Representation "coefficient by coefficient"

- The system is stored as an equation in the big field $\mathbb{F}_{2^m}$.
- Example in $\mathbb{F}_2$: let $\mathbb{F}_4 = \mathbb{F}_2[X]/(\alpha^2 + \alpha + 1)$,
  $\mathbf{p}(x_1, x_2, x_3) = 1 \times (1) + \alpha \times (2)$
  $= (\alpha + 1)x_1x_2 + \alpha x_1x_3 + x_2x_3 + (\alpha + 1)x_1 + 1$
- This representation is used in [Berbain, Billet, Gilbert, Efficient Implementations of Multivariate Quadratic Systems] and `MQsoft`.

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	x_1	x_2	x_3	x_4
x_1	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) =$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	x_1	x_2	x_3	x_4
x_1	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	x_1	x_2	x_3	x_4
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	x_2	x_3	x_4
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	x_2	x_3	x_4
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	x_3	x_4
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	x_3	x_4
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	x_4
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	x_4
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
x_2		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
x_3			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
$x_3 = 1$			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
$x_3 = 1$			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3} + \mathbf{p}_{3,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
$x_3 = 1$			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
x_4				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3} + \mathbf{p}_{3,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
$x_3 = 1$			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
$x_4 = 0$				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3} + \mathbf{p}_{3,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
$x_3 = 1$			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
$x_4 = 0$				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3} + \mathbf{p}_{3,3}$

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
$x_3 = 1$			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
$x_4 = 0$				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3} + \mathbf{p}_{3,3}$
- On average, 75% of the monomials are null.

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
$x_3 = 1$			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
$x_4 = 0$				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3} + \mathbf{p}_{3,3}$
- On average, 75% of the monomials are null.
- `MQsoft`: speed-up of 38%, based on unrolled loops and an Euclidean division of the indices of the loops.

Evaluation in variable-time

- $\mathbf{p} \in \mathbb{F}_{2^m}[x_1, \ldots, x_n]$ is stored as a quadratic form in the row-major order.
- Example:

$\mathbf{p}.cst$	$x_1 = 1$	$x_2 = 0$	$x_3 = 1$	$x_4 = 0$
$x_1 = 1$	$\mathbf{p}_{1,1}$	$\mathbf{p}_{1,2}$	$\mathbf{p}_{1,3}$	$\mathbf{p}_{1,4}$
$x_2 = 0$		$\mathbf{p}_{2,2}$	$\mathbf{p}_{2,3}$	$\mathbf{p}_{2,4}$
$x_3 = 1$			$\mathbf{p}_{3,3}$	$\mathbf{p}_{3,4}$
$x_4 = 0$				$\mathbf{p}_{4,4}$

- $\mathbf{p}(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0) = \mathbf{p}.cst + \mathbf{p}_{1,1} + \mathbf{p}_{1,3} + \mathbf{p}_{3,3}$
- On average, 75% of the monomials are null.
- `MQsoft`: speed-up of 38%, based on unrolled loops and an Euclidean division of the indices of the loops.
- Our constant-time implementation is 10% faster on Skylake, by using the vpermq instruction in a specific way.

Root finding in $\mathbb{F}_{2^n}[X]$

Root finding algorithm of $F \in \mathbb{F}_{2^n}[X]$ [von zur Gathen, Gerhard, Modern Computer Algebra]

1. $H = X^{2^n} - X \bmod F$.
2. $G = \mathtt{GCD}(F, H)$. G is split and has a small number of roots.
3. Computation of all roots of G with an equal-degree factorization algorithm.

Root finding in $\mathbb{F}_{2^n}[X]$

Root finding algorithm of $F \in \mathbb{F}_{2^n}[X]$ [von zur Gathen, Gerhard, Modern Computer Algebra]

1. $H = X^{2^n} - X \bmod F$.
2. $G = \texttt{GCD}(F, H)$. G is split and has a small number of roots.
3. Computation of all roots of G with an equal-degree factorization algorithm.

Specificity of the HFE polynomial F

$$F = \sum_{\substack{0 \leqslant j < i < n \\ 2^i + 2^j \leqslant D}} A_{i,j} X^{2^i + 2^j} + \sum_{\substack{0 \leqslant i < n \\ 2^i \leqslant D}} B_i X^{2^i} + C \in \mathbb{F}_{2^n}[X]$$

- F is sparse (quadratic form, $\frac{1}{2} \log_2(D)^2$ coefficients).

Repeating squaring algorithm

Classical method to compute $X^{2^n} - X \bmod F$.

function RepeatingSquaring($F \in \mathbb{F}_{2^n}[X]$)
    $X_i \leftarrow X$ ▷ X_i is $X^{2^i} \bmod F$
    for i from 1 to n **do**
        $X_i \leftarrow X_i^2 \bmod F$
    end for
    return $X_i + X$
end function

Repeating squaring algorithm

Classical method to compute $X^{2^n} - X \bmod F$.

function RepeatingSquaring($F \in \mathbb{F}_{2^n}[X]$)
  $X_i \leftarrow X$ ▷ X_i is $X^{2^i} \bmod F$
  for i from 1 to n **do**
    $X_i \leftarrow X_i^2 \bmod F$
  end for
  return $X_i + X$
end function

Specificities

- The odd degree terms of X_i^2 are zero.
- Modular reduction by a sparse polynomial: $\frac{D}{2}\log_2(D)^2$ field multiplications.

Improvement of the repeating squaring algorithm

Let:

- $X_i^2 = FQ + X_{i+1}$ the Euclidean division of X_i^2 by F,
- $F = F_{\text{low}} + X^{d+1}F_{\text{high}}$, with $f_d X^d$ the largest odd degree term,
- $Q = Q_{\text{low}} + X^{d-1}Q_{\text{high}}$.

Improvement of the repeating squaring algorithm

Let:

- $X_i^2 = FQ + X_{i+1}$ the Euclidean division of X_i^2 by F,
- $F = F_{\text{low}} + X^{d+1} F_{\text{high}}$, with $f_d X^d$ the largest odd degree term,
- $Q = Q_{\text{low}} + X^{d-1} Q_{\text{high}}$.

We have:

1. The odd degree terms of F_{high} are null,

Improvement of the repeating squaring algorithm

Let:

- $X_i^2 = FQ + X_{i+1}$ the Euclidean division of X_i^2 by F,
- $F = F_{\text{low}} + X^{d+1}F_{\text{high}}$, with $f_d X^d$ the largest odd degree term,
- $Q = Q_{\text{low}} + X^{d-1}Q_{\text{high}}$.

We have:

1. The odd degree terms of F_{high} are null,
2. The odd degree terms of Q_{high} are null,

Improvement of the repeating squaring algorithm

Let:

- $X_i^2 = FQ + X_{i+1}$ the Euclidean division of X_i^2 by F,
- $F = F_{\text{low}} + X^{d+1} F_{\text{high}}$, with $f_d X^d$ the largest odd degree term,
- $Q = Q_{\text{low}} + X^{d-1} Q_{\text{high}}$.

We have:

1. The odd degree terms of F_{high} are null,
2. The odd degree terms of Q_{high} are null,
3. If D is even, $\tilde{F} = F - f_d X^d = \tilde{F}_{\text{low}} + X^{\tilde{d}+1} \tilde{F}_{\text{high}}$ with $\tilde{d} = \frac{d+1}{2}$

Improvement of the repeating squaring algorithm

Let:

- $X_i^2 = FQ + X_{i+1}$ the Euclidean division of X_i^2 by F,
- $F = F_{\text{low}} + X^{d+1}F_{\text{high}}$, with $f_d X^d$ the largest odd degree term,
- $Q = Q_{\text{low}} + X^{d-1}Q_{\text{high}}$.

We have:

1. The odd degree terms of F_{high} are null,
2. The odd degree terms of Q_{high} are null,
3. If D is even, $\tilde{F} = F - f_d X^d = \tilde{F}_{\text{low}} + X^{\tilde{d}+1}\tilde{F}_{\text{high}}$ with $\tilde{d} = \frac{d+1}{2}$

Theorem (simplified)

Let D be an even integer, and F be a D-degree HFE polynomial. By removing s odd degree terms of F, the Euclidean division of X_i by F can be accelerated by a factor < 2.

Sparse HFE polynomials and security

s	d	Number of non-zero terms of Q	Speed-up
0	129	129	0%
1	65	97	33%
2	33	81	59%
3	17	73	77%
4	9	69	87%
5	5	67	93%
6	3	66	95%
7	1	65 (only even degree terms)	98%

Speed-up of the Euclidean division of X_i by F for $D = 130$. We remove $\{f_{129}X^{129}, f_{65}X^{65}, \ldots, f_{2d-1}X^{2d-1}\} = s$ terms.

Sparse HFE polynomials and security

s	d	Number of non-zero terms of Q	Speed-up	$D_{\text{reg}}^{\text{Experimental}}$
0	129	129	0%	5
1	65	97	33%	5
2	33	81	59%	5
3	17	73	77%	5
4	9	69	87%	5
5	5	67	93%	5
6	3	66	95%	5
7	1	65 (only even degree terms)	98%	5

Speed-up of the Euclidean division of X_i by F for $D = 130$. We remove $\{f_{129}X^{129}, f_{65}X^{65}, \ldots, f_{2d-1}X^{2d-1}\} = s$ terms.

Complexity of the Gröbner Basis attack [FauJou03]

The complexity of the direct attack against the HFE-based schemes is $O(n^{\omega D_{\text{reg}}})$, with D_{reg} the degree of regularity and $2 \leq \omega \leq 3$.

Performance

n	D	s	NTL	Magma	MQsoft
174	513	0	1090	-3.6%	**+840%**
	514	3	1100	+46%	**+1500%**
354	513	0	4370	+16%	**+640%**
	514	3	4390	+88%	**+1200%**

Number of mega cycles to find the roots of a HFE polynomial with NTL, followed by the speed-ups obtained respectively with Magma and MQsoft (Skylake processor).

Results

- NTL is not adapted to the sparse polynomials.
- Magma exploits the parameter s with a variable-time implementation.
- MQsoft is fast and has a constant-time sparse repeating squaring algorithm.

Conclusion

Performance

- MQsoft is an efficient C library faster than the generic libraries.
- MQsoft improves the NIST candidates GeMSS and Gui.
- The parameter s accelerates the root finding of HFE polynomials in $\mathbb{F}_{2^n}[X]$.

Perspectives

- The security of the parameter s must be studied in depth.
- To propose methods in constant-time for the GCD and the choice of a root during the root finding.
- To add the use of AVX-512 and the VPCLMULQDQ instruction[a].

[a] Available on the future Ice Lake processors (10^{th} generation)

Questions?

Thank you for your attention.

References I

- Daniel J. Bernstein, Tung Chou and Peter Schwabe. McBits: Fast Constant-Time Code-Based Cryptography. CHES 2013.
- Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian and Tancrède Lepoint. NFLlib: NTT-Based Fast Lattice Library. CT-RSA 2016.
- Tsutomu Matsumoto and Hideki Imai. Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption. EUROCRYPT '88.
- Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. EUROCRYPT '96.
- Jean-Charles Faugère and Antoine Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. CRYPTO '03.
- Manuel Bluhm and Shay Gueron. Fast software implementation of binary elliptic curve cryptography. J. Cryptographic Engineering.
- Côme Berbain, Olivier Billet, Henri Gilbert. Efficient Implementations of Multivariate Quadratic Systems. SAC 2006.

References II

- Ming-Shing Chen, Wen-Ding Li, Bo-Yuan Peng, Bo-Yin Yang and Chen-Mou Cheng. Implementing 128-Bit Secure MPKC Signatures. IEICE Transactions.
- Joachim von zur Gathen and Jürgen Gerhard. Modern Computer Algebra (3. ed).