From 5-pass $\mathcal{MQ}$-based identification to $\mathcal{MQ}$-based signatures

Ming-Shing Chen[1,2], Andreas Hülsing[3], **Joost Rijneveld**[4], Simona Samardjiska[5], Peter Schwabe[4]

National Taiwan University[1] / Academia Sinica[2], Taipei, Taiwan
Technische Universiteit Eindhoven, The Netherlands[3]
Radboud University, Nijmegen, The Netherlands[4]
"Ss. Cyril and Methodius" University, Skopje, Republic of Macedonia[5]

2016-11-18
DiS Lunch Talk

Post-quantum signatures

Problem: we want a post-quantum signature scheme

- Security arguments
- ‘Acceptable’ speed and size

Post-quantum signatures

Problem: we want a post-quantum signature scheme

- Security arguments
- 'Acceptable' speed and size

Solutions:

- Hash-based: SPHINCS, XMSS
  - Slow or stateful
- Lattice-based: (Ring-)TESLA, BLISS, GLP
  - Large keys, or additional structure
- $\mathcal{MQ}$: ?
  - Unclear security: many broken (except HFEv-, UOV)

Overview

$\underline{\mathrm{MQDSS}}$ is a signature scheme

Overview

MQDSS is a signature scheme obtained by applying a Fiat-Shamir transform

Overview

$\underline{\mathrm{MQDSS}}$ is a signature scheme obtained by applying a <u>Fiat-Shamir transform</u> to a 5-pass <u>identification scheme</u>

Overview

$\underline{\mathrm{MQDSS}}$ is a signature scheme obtained by applying a Fiat-Shamir transform to a 5-pass identification scheme which relies on hardness of the $\mathcal{MQ}$ problem.

Overview

MQDSS is a signature scheme obtained by applying a Fiat-Shamir transform to a 5-pass identification scheme which relies on hardness of the $\mathcal{MQ}$ problem.

MQDSS-31-64 provides post-quantum secure signatures.

This work

- Transform 5-pass IDS to signature schemes
- Prove an earlier attempt [EDV+12] vacuous
- Propose MQDSS
- Instantiate as MQDSS-31-64
- Implement using AVX2

But also:

- Reduction in the ROM (not in QROM)
- No tight proof

Identification schemes

$\mathcal{P}$		$\mathcal{V}$
$\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \leftarrow_R \mathsf{ChS}(1^k)$
$\mathsf{resp} \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch})$	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

Identification schemes

$$
\begin{array}{lcl}
\mathcal{P} & & \mathcal{V} \\
\hline
\mathsf{com} \leftarrow \mathcal{P}_0(\mathsf{sk}) & & \\
 & \xrightarrow{\mathsf{com}} & \\
 & & \mathsf{ch} \leftarrow_R \mathsf{ChS}(1^k) \\
 & \xleftarrow{\mathsf{ch}} & \\
\mathsf{resp} \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}) & & \\
 & \xrightarrow{\mathsf{resp}} & \\
 & & b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})
\end{array}
$$

Informally:

1. Prover commits to some (random) value
2. Verifier picks a challenge ‘ch’
3. Prover computes response ‘resp’
4. Verifier checks if response matches challenge

Zero-knowledge

Special Soundness: given pk, $(com, ch, resp)$, $(com, ch', resp')$, find sk.

Honest-Verifier Zero-Knowledge: simulator can 'fake' transcripts.

Zero-knowledge

Special Soundness: given pk, $(com, ch, resp)$, $(com, ch', resp')$, find sk.

- Shows knowledge of secret
- Prover can 'guess right': soundness error κ

Honest-Verifier Zero-Knowledge: simulator can 'fake' transcripts.

- Shows that transcripts do not leak the secret

Zero-knowledge

Special Soundness: given pk, $(com, ch, resp)$, $(com, ch', resp')$, find sk.

- Shows knowledge of secret
- Prover can 'guess right': soundness error κ

Honest-Verifier Zero-Knowledge: simulator can 'fake' transcripts.

- Shows that transcripts do not leak the secret

$\Rightarrow$ Passively secure IDS

Fiat-Shamir transform

- Transform passive IDS into signature
- Non-interactive:

Fiat-Shamir transform

- Transform passive IDS into signature
- Non-interactive:
  - Signer is ‘prover’
  - Function $\mathcal{H}$ provides challenges
  - Transcript is signature

Fiat-Shamir transform

- Transform passive IDS into signature
- Non-interactive:
  - Signer is ‘prover’
  - Function $\mathcal{H}$ provides challenges
  - Transcript is signature
- Repeat, to compensate for error κ

Fiat-Shamir transform

- Transform passive IDS into signature
- Non-interactive:
  - Signer is 'prover'
  - Function $\mathcal{H}$ provides challenges
  - Transcript is signature
- Repeat, to compensate for error κ
- Generalise to 5-pass (of certain form)
  - Reduces knowledge error

Fiat-Shamir transform

Signing a message M (r rounds such that $\kappa^r < \frac{1}{2}^k$):

Verifying a signature σ:

Fiat-Shamir transform

Signing a message M (r rounds such that $\kappa^r < \frac{1}{2}^k$):

1. Generate r commitments $\mathsf{com}_i \leftarrow \mathcal{P}_0(\mathsf{sk})$

Verifying a signature σ:

Fiat-Shamir transform

Signing a message M (r rounds such that $\kappa^r < \frac{1}{2}^k$):

1. Generate r commitments $\mathsf{com}_i \leftarrow \mathcal{P}_0(\mathsf{sk})$
2. Construct $\sigma_0 = \mathsf{com}_0 \| \dots \| \mathsf{com}_{r-1}$

Verifying a signature σ:

Fiat-Shamir transform

Signing a message M (r rounds such that $\kappa^r < \frac{1}{2}^k$):

1. Generate r commitments $\mathsf{com}_i \leftarrow \mathcal{P}_0(\mathsf{sk})$
2. Construct $\sigma_0 = \mathsf{com}_0 \| \ldots \| \mathsf{com}_{r-1}$
3. Generate challenges $\mathsf{ch}_i \leftarrow \mathcal{H}(\sigma_0 \| M)$

Verifying a signature σ:

Fiat-Shamir transform

Signing a message M (r rounds such that $\kappa^r < \frac{1}{2}^k$):

1. Generate r commitments $\mathsf{com}_i \leftarrow \mathcal{P}_0(\mathsf{sk})$
2. Construct $\sigma_0 = \mathsf{com}_0 \| \dots \| \mathsf{com}_{r-1}$
3. Generate challenges $\mathsf{ch}_i \leftarrow \mathcal{H}(\sigma_0 \| M)$
4. Compute $\mathsf{resp}_i \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}_i, \mathsf{ch}_i)$

Verifying a signature σ:

Fiat-Shamir transform

Signing a message M (r rounds such that $\kappa^r < \frac{1}{2}^k$):

1. Generate r commitments $\mathsf{com}_i \leftarrow \mathcal{P}_0(\mathsf{sk})$
2. Construct $\sigma_0 = \mathsf{com}_0 \| \ldots \| \mathsf{com}_{r-1}$
3. Generate challenges $\mathsf{ch}_i \leftarrow \mathcal{H}(\sigma_0 \| M)$
4. Compute $\mathsf{resp}_i \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}_i, \mathsf{ch}_i)$
5. Construct $\sigma_1 = (\mathsf{resp}_0, \| \ldots \| \mathsf{resp}_{r-1})$, $\sigma = (\sigma_0, \sigma_1)$

Verifying a signature σ:

Fiat-Shamir transform

Signing a message M (r rounds such that $\kappa^r < \frac{1}{2}^k$):

1. Generate r commitments $\mathsf{com}_i \leftarrow \mathcal{P}_0(\mathsf{sk})$
2. Construct $\sigma_0 = \mathsf{com}_0 \| \dots \| \mathsf{com}_{r-1}$
3. Generate challenges $\mathsf{ch}_i \leftarrow \mathcal{H}(\sigma_0 \| M)$
4. Compute $\mathsf{resp}_i \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}_i, \mathsf{ch}_i)$
5. Construct $\sigma_1 = (\mathsf{resp}_0, \| \dots \| \mathsf{resp}_{r-1})$, $\sigma = (\sigma_0, \sigma_1)$

Verifying a signature σ:

1. Generate challenges $\mathsf{ch}_i \leftarrow \mathcal{H}(\sigma_0 \| M)$

Fiat-Shamir transform

Signing a message M (r rounds such that $\kappa^r < \frac{1}{2}^k$):

1. Generate r commitments $\mathsf{com}_i \leftarrow \mathcal{P}_0(\mathsf{sk})$
2. Construct $\sigma_0 = \mathsf{com}_0 \| \ldots \| \mathsf{com}_{r-1}$
3. Generate challenges $\mathsf{ch}_i \leftarrow \mathcal{H}(\sigma_0 \| M)$
4. Compute $\mathsf{resp}_i \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}_i, \mathsf{ch}_i)$
5. Construct $\sigma_1 = (\mathsf{resp}_0, \| \ldots \| \mathsf{resp}_{r-1})$, $\sigma = (\sigma_0, \sigma_1)$

Verifying a signature σ:

1. Generate challenges $\mathsf{ch}_i \leftarrow \mathcal{H}(\sigma_0 \| M)$
2. Verify that $\forall i, \mathsf{Vf}(\mathsf{pk}, \mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)$

$\mathcal{MQ}$ problem

The function family $\mathcal{MQ}(n, m, \mathbb{F}_q)$:

$$\left\{ \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) \middle| \; f_l(\mathbf{x}) = \begin{array}{l} \sum_{i,j} a_{l,i,j} x_i x_j + \sum_i b_{l,i}, \\ \text{for } a_{l,i,j},\ b_{l,i} \in \mathbb{F}_q \\ l \in 1, \ldots, m \end{array} \right\}$$

$\mathcal{MQ}$ problem

The function family $\mathcal{MQ}(n, m, \mathbb{F}_q)$:

$$\left\{ \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) \middle| \, f_l(\mathbf{x}) = \begin{array}{l} \sum_{i,j} a_{l,i,j} x_i x_j + \sum_i b_{l,i}, \\ \text{for } a_{l,i,j},\ b_{l,i} \in \mathbb{F}_q \\ l \in 1, \ldots, m \end{array} \right\}$$

Problem: For given $\mathbf{y} \in \mathbb{F}_q^m$, find $\mathbf{x} \in \mathbb{F}_q^n$ such that $\mathbf{F}(\mathbf{x}) = \mathbf{y}$.

$\mathcal{MQ}$ problem

The function family $\mathcal{MQ}(n, m, \mathbb{F}_q)$:

$$\left\{ \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) \middle| \; f_l(\mathbf{x}) = \begin{array}{l} \sum_{i,j} a_{l,i,j} x_i x_j + \sum_i b_{l,i}, \\ \text{for } a_{l,i,j},\ b_{l,i} \in \mathbb{F}_q \\ l \in 1, \ldots, m \end{array} \right\}$$

Problem: For given $\mathbf{y} \in \mathbb{F}_q^m$, find $\mathbf{x} \in \mathbb{F}_q^n$ such that $\mathbf{F}(\mathbf{x}) = \mathbf{y}$.

i.e., solve the system of equations:

$$\begin{aligned} f_0(\mathbf{x}) =& a_{0,0,0} x_0 x_0 + a_{0,0,1} x_0 x_1 + \ldots + a_{0,n,n} x_n x_n + b_{0,0} + \ldots + b_{0,n} \\ &\vdots \\ f_m(\mathbf{x}) =& a_{m,0,0} x_0 x_0 + a_{m,0,1} x_0 x_1 + \ldots + a_{m,n,n} x_n x_n + b_{m,0} + \ldots + b_{m,n} \end{aligned}$$

Sakumoto et al. 5-pass IDS [SSH11]

- Key technique: $\mathbf{v} = \mathbf{F}(\mathbf{s}) = \mathbf{F}(\mathbf{r_0}) + \mathbf{F}(\mathbf{r_1}) + \mathbf{G}(\mathbf{r_0}, \mathbf{r_1})$

$$
\begin{array}{lcl}
\mathcal{P} : (\mathbf{F}, \mathbf{v}, \mathbf{s}) & & \mathcal{V} : (\mathbf{F}, \mathbf{v}) \\
\hline
\mathbf{r}_0, \mathbf{t}_0 \leftarrow_R \mathbb{F}_q^n, \mathbf{e}_0 \leftarrow_R \mathbb{F}_q^m & & \\
\mathbf{r}_1 \leftarrow \mathbf{s} - \mathbf{r}_0 & & \\
c_0 \leftarrow Com(\mathbf{r}_0, \mathbf{t}_0, \mathbf{e}_0) & & \\
c_1 \leftarrow Com(\mathbf{r}_1, \mathbf{G}(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{e}_0) & \xrightarrow{(c_0, c_1)} & \\
 & \xleftarrow{\alpha} & \alpha \leftarrow_R \mathbb{F}_q \\
\mathbf{t}_1 \leftarrow \alpha \mathbf{r}_0 - \mathbf{t}_0 & & \\
\mathbf{e}_1 \leftarrow \alpha \mathbf{F}(\mathbf{r}_0) - \mathbf{e}_0 & \xrightarrow{\mathrm{resp}_1 = (\mathbf{t}_1, \mathbf{e}_1)} & \\
 & \xleftarrow{\mathrm{ch}_2} & \mathrm{ch}_2 \leftarrow_R \{0, 1\} \\
\text{If } \mathrm{ch}_2 = 0,\ \mathrm{resp}_2 \leftarrow \mathbf{r}_0 & & \\
\text{Else } \mathrm{resp}_2 \leftarrow \mathbf{r}_1 & \xrightarrow{\mathrm{resp}_2} & \\
 & & \text{If } \mathrm{ch}_2 = 0,\ \text{Parse } \mathrm{resp}_2 = \mathbf{r}_0,\ \text{check} \\
 & & c_0 \stackrel{?}{=} Com(\mathbf{r}_0, \alpha \mathbf{r}_0 - \mathbf{t}_1, \alpha \mathbf{F}(\mathbf{r}_0) - \mathbf{e}_1) \\
 & & \text{Else Parse } \mathrm{resp}_2 = \mathbf{r}_1,\ \text{check} \\
 & & c_1 \stackrel{?}{=} Com(\mathbf{r}_1, \alpha(\mathbf{v} - \mathbf{F}(\mathbf{r}_1)) - \mathbf{G}(\mathbf{t}_1, \mathbf{r}_1) - \mathbf{e}_1)
\end{array}
$$

MQDSS

- Generate keys
  - Sample seeds $\in \{0,1\}^k$, $\mathbf{sk} \in \mathbb{F}_q$
  - Expand to $\mathbf{F}$, compute $\mathbf{pk} = \mathbf{F}(\mathbf{sk})$

MQDSS

- Generate keys
  - Sample seeds $\in \{0,1\}^k$, **sk** $\in \mathbb{F}_q$
  - Expand to **F**, compute $\mathbf{pk} = \mathbf{F}(\mathbf{sk})$
- Signing
  - Sign deterministic digest D over M

MQDSS

- Generate keys
  - Sample seeds $\in \{0,1\}^k$, $\mathbf{sk} \in \mathbb{F}_q$
  - Expand to $\mathbf{F}$, compute $\mathbf{pk} = \mathbf{F}(\mathbf{sk})$
- Signing
  - Sign deterministic digest D over M
  - Perform r rounds of transformed IDS
    - $2r$ commitments
    - $2r$ $\mathcal{MQ}$ evaluations
    - Some multiplications in $\mathbb{F}_q$

MQDSS

- Generate keys
  - Sample seeds $\in \{0,1\}^k$, $\mathbf{sk} \in \mathbb{F}_q$
  - Expand to $\mathbf{F}$, compute $\mathbf{pk} = \mathbf{F}(\mathbf{sk})$
- Signing
  - Sign deterministic digest D over M
  - Perform r rounds of transformed IDS
    - $2r$ commitments
    - $2r$ $\mathcal{MQ}$ evaluations
    - Some multiplications in $\mathbb{F}_q$
  - Tricks to reduce size
    - Only include necessary commits (hash others) [SSH11]
    - Commit to seeds

MQDSS

- Generate keys
  - Sample seeds $\in \{0,1\}^k$, $\mathbf{sk} \in \mathbb{F}_q$
  - Expand to $\mathbf{F}$, compute $\mathbf{pk} = \mathbf{F}(\mathbf{sk})$
- Signing
  - Sign deterministic digest D over M
  - Perform r rounds of transformed IDS
    - $2r$ commitments
    - $2r$ $\mathcal{MQ}$ evaluations
    - Some multiplications in $\mathbb{F}_q$
  - Tricks to reduce size
    - Only include necessary commits (hash others) [SSH11]
    - Commit to seeds
- Verifying
  - Reconstruct D, $\mathbf{F}$

MQDSS

- Generate keys
  - Sample seeds $\in \{0,1\}^k$, $\mathbf{sk} \in \mathbb{F}_q$
  - Expand to $\mathbf{F}$, compute $\mathbf{pk} = \mathbf{F}(\mathbf{sk})$
- Signing
  - Sign deterministic digest D over M
  - Perform r rounds of transformed IDS
    - $2r$ commitments
    - $2r$ $\mathcal{MQ}$ evaluations
    - Some multiplications in $\mathbb{F}_q$
  - Tricks to reduce size
    - Only include necessary commits (hash others) [SSH11]
    - Commit to seeds
- Verifying
  - Reconstruct D, $\mathbf{F}$
  - Reconstruct challenges from σ_0, σ_1
  - Verify responses in σ_2

MQDSS

- Generate keys
  - Sample seeds $\in \{0,1\}^k$, $\mathbf{sk} \in \mathbb{F}_q$
  - Expand to $\mathbf{F}$, compute $\mathbf{pk} = \mathbf{F}(\mathbf{sk})$
- Signing
  - Sign deterministic digest D over M
  - Perform r rounds of transformed IDS
    - $2r$ commitments
    - $2r$ $\mathcal{MQ}$ evaluations
    - Some multiplications in $\mathbb{F}_q$
  - Tricks to reduce size
    - Only include necessary commits (hash others) [SSH11]
    - Commit to seeds
- Verifying
  - Reconstruct D, $\mathbf{F}$
  - Reconstruct challenges from σ_0, σ_1
  - Verify responses in σ_2
  - Reconstruct missing commitments
  - Check combined commitments hash

MQDSS

- Generate keys
  - Sample seeds $\in \{0, 1\}^k$, $\mathbf{sk} \in \mathbb{F}_q$
  - Expand to $\mathbf{F}$, compute $\mathbf{pk} = \mathbf{F}(\mathbf{sk})$
- Signing
  - Sign deterministic digest D over M
  - Perform r rounds of transformed IDS
    - $2r$ commitments
    - $2r$ $\mathcal{MQ}$ evaluations
    - Some multiplications in $\mathbb{F}_q$
  - Tricks to reduce size
    - Only include necessary commits (hash others) [SSH11]
    - Commit to seeds
- Verifying
  - Reconstruct D, $\mathbf{F}$
  - Reconstruct challenges from σ_0, σ_1
  - Verify responses in σ_2
  - Reconstruct missing commitments
  - Check combined commitments hash
- Parameters: k, n, m, $\mathbb{F}_q$, Com, hash functions, PRGs

MQDSS-31-64

- Security parameter $k = 256$
- Soundness error κ depends on q
  - $\kappa = \frac{q+1}{2q}$
  - Determines number of rounds: $r = 269$, $\kappa^{269} < \frac{1}{2}^{256}$
- $\mathbb{F}_q = \mathbb{F}_{31}$, $n = m = 64$
  - Bounded by attacks
  - Chosen for ease of implementation

MQDSS-31-64

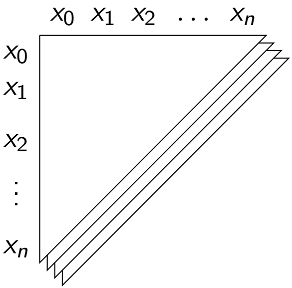
- Security parameter $k = 256$
- Soundness error κ depends on q
  - $\kappa = \frac{q+1}{2q}$
  - Determines number of rounds: $r = 269$, $\kappa^{269} < \frac{1}{2}^{256}$
- $\mathbb{F}_q = \mathbb{F}_{31}$, $n = m = 64$
  - Bounded by attacks
  - Chosen for ease of implementation
- Commitments, hashes, PRGs: SHA3-256, SHAKE-128

Evaluating $\mathcal{MQ}$

- From $\mathbf{F}(\mathbf{x})$ to $\mathbf{x}$ is hard
- From $\mathbf{x}$ to $\mathbf{F}(\mathbf{x})$ should be easy

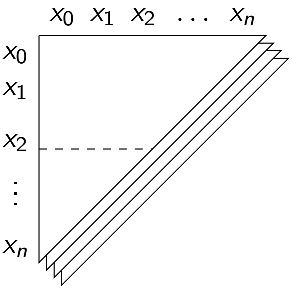
Evaluating $\mathcal{MQ}$

- From $\mathbf{F}(\mathbf{x})$ to $\mathbf{x}$ is hard
- From $\mathbf{x}$ to $\mathbf{F}(\mathbf{x})$ should be fast



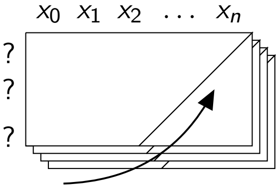
Evaluating $\mathcal{MQ}$

- From $\mathbf{F}(\mathbf{x})$ to $\mathbf{x}$ is hard
- From $\mathbf{x}$ to $\mathbf{F}(\mathbf{x})$ should be fast



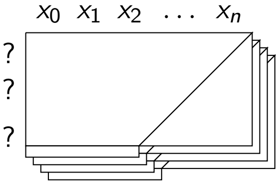
Evaluating $\mathcal{MQ}$

- From $\mathbf{F}(\mathbf{x})$ to $\mathbf{x}$ is hard
- From $\mathbf{x}$ to $\mathbf{F}(\mathbf{x})$ should be fast



Evaluating $\mathcal{MQ}$

- From $\mathbf{F}(\mathbf{x})$ to $\mathbf{x}$ is hard
- From $\mathbf{x}$ to $\mathbf{F}(\mathbf{x})$ should be fast



Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

$\boxed{0}$	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

$<<<$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

$<<<$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F
-	-	-	-	-	-	-	-	88	99	AA	BB	8C	9D	AE	BF

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

$<<<$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F
-	-	-	-	-	-	-	-	88	99	AA	BB	8C	9D	AE	BF
90	A1	B2	83	94	A5	B6	87	98	A9	BA	8B	9C	AD	BE	8F

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F
-	-	-	-	-	-	-	-	88	99	AA	BB	8C	9D	AE	BF
90	A1	B2	83	94	A5	B6	87	98	A9	BA	8B	9C	AD	BE	8F
-	-	-	-	-	-	-	-	-	-	-	-	CC	DD	EE	FF

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

$<<<$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F
-	-	-	-	-	-	-	-	88	99	AA	BB	8C	9D	AE	BF
90	A1	B2	83	94	A5	B6	87	98	A9	BA	8B	9C	AD	BE	8F
-	-	-	-	-	-	-	-	-	-	-	-	CC	DD	EE	FF
D0	E1	F2	C3	D4	E5	F6	C7	D8	E9	FA	CB	DC	ED	FE	CF

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F
-	-	-	-	-	-	-	-	88	99	AA	BB	8C	9D	AE	BF
90	A1	B2	83	94	A5	B6	87	98	A9	BA	8B	9C	AD	BE	8F
-	-	-	-	-	-	-	-	-	-	-	-	CC	DD	EE	FF
D0	E1	F2	C3	D4	E5	F6	C7	D8	E9	FA	CB	DC	ED	FE	CF
02	13	-	-	42	53	-	-	82	93	-	-	C2	D3	-	-

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F
-	-	-	-	-	-	-	-	88	99	AA	BB	8C	9D	AE	BF
90	A1	B2	83	94	A5	B6	87	98	A9	BA	8B	9C	AD	BE	8F
-	-	-	-	-	-	-	-	-	-	-	-	CC	DD	EE	FF
D0	E1	F2	C3	D4	E5	F6	C7	D8	E9	FA	CB	DC	ED	FE	CF
02	13	-	-	42	53	-	-	82	93	-	-	C2	D3	-	-
06	17	-	-	46	57	-	-	86	97	-	-	C6	D7	-	-

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F
-	-	-	-	-	-	-	-	88	99	AA	BB	8C	9D	AE	BF
90	A1	B2	83	94	A5	B6	87	98	A9	BA	8B	9C	AD	BE	8F
-	-	-	-	-	-	-	-	-	-	-	-	CC	DD	EE	FF
D0	E1	F2	C3	D4	E5	F6	C7	D8	E9	FA	CB	DC	ED	FE	CF
02	13	-	-	42	53	-	-	82	93	-	-	C2	D3	-	-
06	17	-	-	46	57	-	-	86	97	-	-	C6	D7	-	-
0A	1B	-	-	4A	5B	-	-	8A	9B	-	-	CA	DB	-	-

Evaluating $\mathcal{MQ}$

- First compute monomials, then evaluate polynomials
- 64 elements in $\mathbb{F}_{31}$; 16 (or 32) per 256 bit AVX2 register
- Monomials: intuition of arrangement using 4×4:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	04	15	26	37	08	19	2A	3B	0C	1D	2E	3F
10	21	32	03	14	25	36	07	18	29	3A	0B	1C	2D	3E	0F
-	-	-	-	44	55	66	77	48	59	6A	7B	4C	5D	6E	7F
50	61	72	43	54	65	76	47	58	69	7A	4B	5C	6D	7E	4F
-	-	-	-	-	-	-	-	88	99	AA	BB	8C	9D	AE	BF
90	A1	B2	83	94	A5	B6	87	98	A9	BA	8B	9C	AD	BE	8F
-	-	-	-	-	-	-	-	-	-	-	-	CC	DD	EE	FF
D0	E1	F2	C3	D4	E5	F6	C7	D8	E9	FA	CB	DC	ED	FE	CF
02	13	-	-	42	53	-	-	82	93	-	-	C2	D3	-	-
06	17	-	-	46	57	-	-	86	97	-	-	C6	D7	-	-
0A	1B	-	-	4A	5B	-	-	8A	9B	-	-	CA	DB	-	-
0E	1F	-	-	4E	5F	-	-	8E	9F	-	-	CE	DF	-	-

Benchmarks & conclusion

- Signatures: ~40 KB ($\approx$ SPHINCS)
- Public and private keys: 72 resp. 64 bytes
- Signing time: ~8.5M cycles (2.43ms)
  - Verification 5.2M, key generation 1.8M
- ~6x faster than SPHINCS, $>$10x slower than lattices

Benchmarks & conclusion

- Signatures: $\sim$40 KB ($\approx$ SPHINCS)
- Public and private keys: 72 resp. 64 bytes
- Signing time: $\sim$8.5M cycles (2.43ms)
  - Verification 5.2M, key generation 1.8M
- $\sim$6x faster than SPHINCS, $>$10x slower than lattices

- Competitive signatures with (non-tight) reduction to $\mathcal{MQ}$

References

Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari.
Public-key identification schemes based on multivariate quadratic polynomials.
In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *LNCS*, pages 706-723. Springer, 2011.

Sidi Mohamed El Yousfi Alaoui, Özgür Dagdelen, Pascal Véron, David Galindo, and Pierre-Louis Cayrel.
Extended security arguments for signature schemes.
In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *Progress in Cryptology – AFRICACRYPT 2012*, volume 7374 of *LNCS*, pages 19-34. Springer, 2012.