

MQDSS

Ming-Shing Chen¹, **Andreas Hülsing**², Joost Rijneveld³, Simona Samardjiska³,
and Peter Schwabe³

- ¹ National Taiwan University / Academia Sinica, Taipei, Taiwan
² Technische Universiteit Eindhoven, Eindhoven, The Netherlands
³ Radboud University, Nijmegen, The Netherlands

2019-08-23

Second NIST PQC Standardization Conference

In a nutshell..

- ▶ \mathcal{MQ} -based 5-pass identification scheme
 - ▶ Fiat-Shamir transform
- ▶ Loose reduction from (only!) \mathcal{MQ} problem
 - ▶ Security proof, instead of typical 'break and tweak' in \mathcal{MQ} cryptography
- ▶ Very small keys, big signatures

In a nutshell..

- ▶ \mathcal{MQ} -based 5-pass identification scheme
 - ▶ Fiat-Shamir transform
- ▶ Loose reduction from (only!) \mathcal{MQ} problem
 - ▶ Security proof, instead of typical 'break and tweak' in \mathcal{MQ} cryptography
- ▶ Very small keys, big signatures

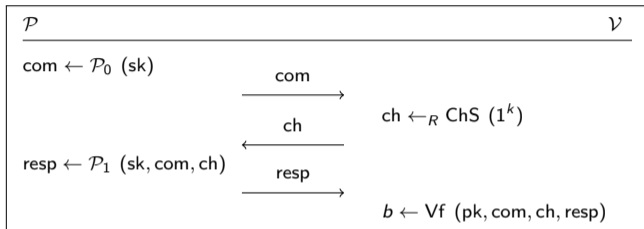
- ▶ First proposed at ASIACRYPT 2016 [CHR⁺16]

In a nutshell..

- ▶ \mathcal{MQ} -based 5-pass identification scheme
 - ▶ Fiat-Shamir transform
- ▶ Loose reduction from (only!) \mathcal{MQ} problem
 - ▶ Security proof, instead of typical 'break and tweak' in \mathcal{MQ} cryptography
- ▶ Very small keys, big signatures
- ▶ First proposed at ASIACRYPT 2016 [CHR⁺16]
- ▶ **Changes in Second Round submission**
 - ▶ Reduction of number of rounds
 - ▶ Added randomness in commitments
 - ▶ More precise analysis of best attacks against \mathcal{MQ}

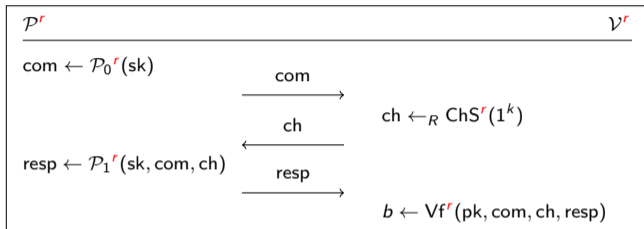
Fiat-Shamir transform

IDS



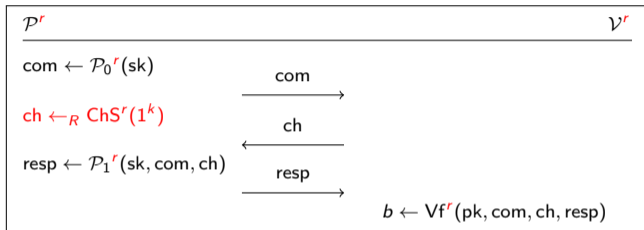
Fiat-Shamir transform

IDS

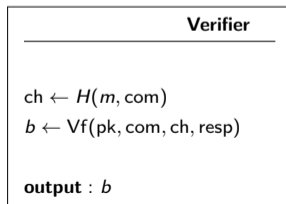
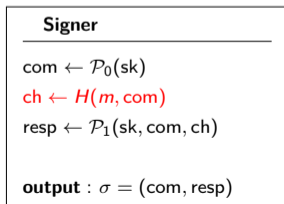


Fiat-Shamir transform

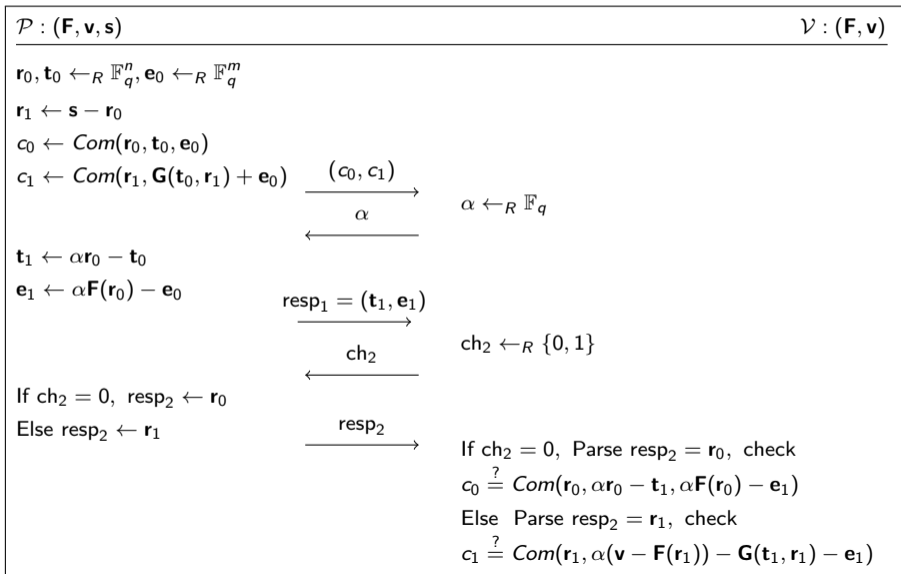
IDS



FS signature



Sakumoto-Shirai-Hiwatari 5-pass IDS [SSH11]



MQDSS

- ▶ Generate keys

- ▶ Sample seed $\mathcal{S}_F \in \{0, 1\}^k$, $\mathbf{s} \in \mathbb{F}_q^n$ $\Rightarrow \mathbf{sk} = (\mathcal{S}_F, \mathbf{s})$

- ▶ Expand \mathcal{S}_F to \mathbf{F} , compute $\mathbf{v} = \mathbf{F}(\mathbf{s})$ $\Rightarrow \mathbf{pk} = (\mathcal{S}_F, \mathbf{v})$

MQDSS

- ▶ Generate keys
 - ▶ Sample seed $\mathcal{S}_F \in \{0, 1\}^k$, $\mathbf{s} \in \mathbb{F}_q^n \quad \Rightarrow \mathbf{sk} = (\mathcal{S}_F, \mathbf{s})$
 - ▶ Expand \mathcal{S}_F to \mathbf{F} , compute $\mathbf{v} = \mathbf{F}(\mathbf{s}) \quad \Rightarrow \mathbf{pk} = (\mathcal{S}_F, \mathbf{v})$
- ▶ Signing
 - ▶ Sign randomized digest D of message M
 - ▶ Perform r parallel rounds of transformed IDS

MQDSS

- ▶ Generate keys
 - ▶ Sample seed $\mathcal{S}_F \in \{0, 1\}^k$, $\mathbf{s} \in \mathbb{F}_q^n \quad \Rightarrow \mathbf{sk} = (\mathcal{S}_F, \mathbf{s})$
 - ▶ Expand \mathcal{S}_F to \mathbf{F} , compute $\mathbf{v} = \mathbf{F}(\mathbf{s}) \quad \Rightarrow \mathbf{pk} = (\mathcal{S}_F, \mathbf{v})$
- ▶ Signing
 - ▶ Sign randomized digest D of message M
 - ▶ Perform r parallel rounds of transformed IDS
- ▶ Verifying
 - ▶ Reconstruct D , \mathbf{F}
 - ▶ Reconstruct challenges
 - ▶ Reconstruct commitments
 - ▶ Check combined commitments hash

MQDSS

- ▶ Generate keys
 - ▶ Sample seed $\mathcal{S}_F \in \{0, 1\}^k$, $\mathbf{s} \in \mathbb{F}_q^n \quad \Rightarrow \mathbf{sk} = (\mathcal{S}_F, \mathbf{s})$
 - ▶ Expand \mathcal{S}_F to \mathbf{F} , compute $\mathbf{v} = \mathbf{F}(\mathbf{s}) \quad \Rightarrow \mathbf{pk} = (\mathcal{S}_F, \mathbf{v})$
- ▶ Signing
 - ▶ Sign randomized digest D of message M
 - ▶ Perform r parallel rounds of transformed IDS
- ▶ Verifying
 - ▶ Reconstruct D , \mathbf{F}
 - ▶ Reconstruct challenges
 - ▶ Reconstruct commitments
 - ▶ Check combined commitments hash
- ▶ Parameters: n, m, q, r (and Com, Hash & PRG)

Round 2 update: Parameter Sets

	Sec. cat.	q	n (= m)	r	pk (bytes)	sk (bytes)	Signature (bytes)
MQDSS-31-48 (Round 1)	1-2	31	48	135 269	46 62	16 32	20854 32882
MQDSS-31-64 (Round 1)	3-4	31	64	202 403	64 88	24 48	43728 67800

Table: Round 1 parameters in black, Round 2 parameters in red.

- ▶ q , $n = m$ chosen using best attacks on \mathcal{MQ}
 - ▶ q additionally chosen for fast arithmetic

Round 2 update: Parameter Sets

	Sec. cat.	q	n ($= m$)	r	pk (bytes)	sk (bytes)	Signature (bytes)
MQDSS-31-48 (Round 1)	1-2	31	48	135 269	46 62	16 32	20854 32882
MQDSS-31-64 (Round 1)	3-4	31	64	202 403	64 88	24 48	43728 67800

Table: Round 1 parameters in black, Round 2 parameters in red.

- ▶ q , $n = m$ chosen using best attacks on \mathcal{MQ}
 - ▶ q additionally chosen for fast arithmetic
- ▶ r chosen such that $2^{-(r \log \frac{2q}{q+1})} < 2^{-k}$
 - ▶ **mistake in calculation in Round 1, chose k too large**

Round 2 update: Commitments

- ▶ MQDSS uses hash for commitments - instantiated with SHAKE-256

Round 2 update: Commitments

- ▶ MQDSS uses hash for commitments - instantiated with SHAKE-256
- ▶ In Round 1 proof assumes *statistically hiding commitments*

Round 2 update: Commitments

- ▶ MQDSS uses hash for commitments - instantiated with SHAKE-256
- ▶ In Round 1 proof assumes *statistically hiding commitments*
 - ▶ Requires a lot of randomness: $5\times$ commitment length [Lei18]
 - ▶ Round 1 MQDSS does not provide any (dedicated) randomness

Round 2 update: Commitments

- ▶ MQDSS uses hash for commitments - instantiated with SHAKE-256
- ▶ In Round 1 proof assumes *statistically hiding commitments*
 - ▶ Requires a lot of randomness: $5 \times$ commitment length [Lei18]
 - ▶ Round 1 MQDSS does not provide any (dedicated) randomness
- ▶ **Round 2:**
 - ▶ Computationally hiding commitments suffices!
 - ▶ Proof updated accordingly

Round 2 update: Commitments

- ▶ MQDSS uses hash for commitments - instantiated with SHAKE-256
- ▶ In Round 1 proof assumes *statistically hiding commitments*
 - ▶ Requires a lot of randomness: $5 \times$ commitment length [Lei18]
 - ▶ Round 1 MQDSS does not provide any (dedicated) randomness
- ▶ **Round 2:**
 - ▶ Computationally hiding commitments suffices!
 - ▶ Proof updated accordingly
 - ▶ Still needs randomness ($2 \times$ commitment length [Lei18])
 - ▶ \Rightarrow adds approx 4KB (10KB) to signature for MQDSS-31-48 (MQDSS-31-64)

Round 2 performance

► Reference implementation

	keygen	signing	verification
MQDSS-31-48	1 192 984	26 630 590	19 840 136
Round 1	1 206 730	52 466 398	38 686 506
MQDSS-31-64	2 767 384	85 268 712	62 306 098
Round 1	2 806 750	169 298 364	123 239 874

Table: Round 1 performance in black, Round 2 performance in red.

Round 2 performance

► Reference implementation

	keygen	signing	verification
MQDSS-31-48	1 192 984	26 630 590	19 840 136
Round 1	1 206 730	52 466 398	38 686 506
MQDSS-31-64	2 767 384	85 268 712	62 306 098
Round 1	2 806 750	169 298 364	123 239 874

Table: Round 1 performance in black, Round 2 performance in red.

► AVX2 implementation (only round 2)

	keygen	signing	verification
MQDSS-31-48	1 074 644	3 816 106	2 551 270
MQDSS-31-64	2 491 050	9 047 148	6 132 948

Round 2 update: More precise analysis of hardness of \mathcal{MQ}

- ▶ Best strategy: Algebraic techniques with exhaustive search
 - ▶ HybridF5 [BFS15], BooleanSolve [BFSS13], Crossbred [JV17]

Round 2 update: More precise analysis of hardness of \mathcal{MQ}

- ▶ Best strategy: Algebraic techniques with exhaustive search
 - ▶ HybridF5 [BFS15], BooleanSolve [BFSS13], Crossbred [JV17]
- ▶ Analyze both classically and using Grover
 - ▶ Classical gates, quantum gates, circuit depth

Round 2 update: More precise analysis of hardness of \mathcal{MQ}

- ▶ Best strategy: Algebraic techniques with exhaustive search
 - ▶ HybridF5 [BFS15], BooleanSolve [BFSS13], Crossbred [JV17]
- ▶ Analyze both classically and using Grover
 - ▶ Classical gates, quantum gates, circuit depth
 - ▶ minor changes in **Round 2** - more precise analysis
 - ▶ no influence to security of parameter sets

Recent attack

- ▶ August 2019, Daniel Kales and Greg Zaverucha - forgery in approx. 2^{95} hash calls for MQDSS-31-48

Recent attack

- ▶ August 2019, Daniel Kales and Greg Zaverucha - forgery in approx. 2^{95} hash calls for MQDSS-31-48
- ▶ Can be mitigated by $\approx 1.4 \times (\text{number of rounds})$

Recent attack

- ▶ August 2019, Daniel Kales and Greg Zaverucha - forgery in approx. 2^{95} hash calls for MQDSS-31-48
- ▶ Can be mitigated by $\approx 1.4 \times (\text{number of rounds})$
- ▶ **Proof still valid!**
 - ▶ Attack is result of not taking into account non-tightness of proof for choosing parameters

Recent attack

- ▶ August 2019, Daniel Kales and Greg Zaverucha - forgery in approx. 2^{95} hash calls for MQDSS-31-48
- ▶ Can be mitigated by $\approx 1.4 \times$ (number of rounds)
- ▶ **Proof still valid!**
 - ▶ Attack is result of not taking into account non-tightness of proof for choosing parameters
- ▶ **New parameters after attack (estimate):**

	Sec. cat.	q	n	r	pk	sk	Signature
MQDSS-31-48 (new) Round 1	1-2	31	48	184 269	46B 62B	16B 32B	28400B 32882B
MQDSS-31-64 (new) Round 1	3-4	31	64	277 403	64B 88B	24B 48B	59928B 67800B

Table: Round 1 parameters in black, New parameters (attack fixed) in red.

Conclusion

- ▶ Fiat-Shamir transform from \mathcal{MQ} -based 5-pass identification scheme
- ▶ Security proof in ROM, instead of typical 'break and tweak' in \mathcal{MQ} cryptography
- ▶ Very small keys, big signatures

Conclusion

- ▶ Fiat-Shamir transform from \mathcal{MQ} -based 5-pass identification scheme
- ▶ Security proof in ROM, instead of typical 'break and tweak' in \mathcal{MQ} cryptography
- ▶ Very small keys, big signatures
- ▶ **Main improvement in Round 2: Smaller signatures**
 - ▶ Even after recent attack & added randomness in commitments

	Sec. cat.	q	n	r	pk	sk	Signature
MQDSS-31-48 (new) Round 1	1-2	31	48	184 269	46B 62B	16B 32B	28400B 32882B
MQDSS-31-64 (new) Round 1	3-4	31	64	277 403	64B 88B	24B 48B	59928B 67800B

Table: Round 1 parameters in black, New parameters (attack fixed) in red.

Conclusion


- ▶ Fiat-Shamir transform from \mathcal{MQ} -based 5-pass identification scheme
- ▶ Security proof in ROM, instead of typical 'break and tweak' in \mathcal{MQ} cryptography
- ▶ Very small keys, big signatures
- ▶ **Main improvement in Round 2: Smaller signatures**
 - ▶ Even after recent attack & added randomness in commitments

	Sec. cat.	q	n	r	pk	sk	Signature
MQDSS-31-48 (new) Round 1	1-2	31	48	184 269	46B 62B	16B 32B	28400B 32882B
MQDSS-31-64 (new) Round 1	3-4	31	64	277 403	64B 88B	24B 48B	59928B 67800B




Table: Round 1 parameters in black, New parameters (attack fixed) in red.

Thank you for your attention!

References I

-  Magali Bardet, Jean-Charles Faugère, and Bruno Salvy.
On the complexity of the F5 Gröbner basis algorithm.
Journal of Symbolic Computation, 70(Supplement C):49 – 70, 2015.
-  Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer.
On the complexity of solving quadratic boolean systems.
Journal of Complexity, 29(1):53–75, 2013.
www-polsys.lip6.fr/~jcf/Papers/BFSS12.pdf.
-  Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe.
From 5-pass \mathcal{MQ} -based identification to \mathcal{MQ} -based signatures.
In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, volume 10032 of *LNCS*, pages 135–165. Springer, 2016.
<http://eprint.iacr.org/2016/708>.

References II

-  Antoine Joux and Vanessa Vitse.
A crossbred algorithm for solving boolean polynomial systems.
Cryptology ePrint Archive, Report 2017/372, 2017.
<http://eprint.iacr.org/2017/372>.
-  Dominik Leichtle.
Post-quantum signatures from identification schemes.
Master Thesis, Technische Universiteit Eindhoven, 2018.
-  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.
<https://www.iacr.org/archive/crypto2011/68410703/68410703.pdf>.

Implementation considerations

- ▶ Very natural internal parallelism

Implementation considerations

- ▶ Very natural internal parallelism
- ▶ Naively constant-time

Implementation considerations

- ▶ Very natural internal parallelism
- ▶ Naively constant-time
- ▶ Mathematically straight-forward
 - ▶ Multiplications and additions in \mathbb{F}_{31}

Implementation considerations

- ▶ Very natural internal parallelism
- ▶ Naively constant-time
- ▶ Mathematically straight-forward
 - ▶ Multiplications and additions in \mathbb{F}_{31}
- ▶ Naively slow
 - ▶ But still constant-time when optimized

Implementation considerations

- ▶ Very natural internal parallelism
- ▶ Naively constant-time
- ▶ Mathematically straight-forward
 - ▶ Multiplications and additions in \mathbb{F}_{31}

- ▶ Naively slow
 - ▶ But still constant-time when optimized
- ▶ Expanding **F** is memory-intensive (134 KiB)
 - ▶ Problematic on small devices

Implementation considerations

- ▶ Very natural internal parallelism
- ▶ Naively constant-time
- ▶ Mathematically straight-forward
 - ▶ Multiplications and additions in \mathbb{F}_{31}

- ▶ Naively slow
 - ▶ But still constant-time when optimized
- ▶ Expanding **F** is memory-intensive (134 KiB)
 - ▶ Problematic on small devices