Hypercube SDitH: a geometric share aggregation approach for more efficient MPCitH Zero Knowledge Proofs and Digital Signatures

C. Aguilar-Melchor, **N. Gama**, J. Howe, A. Hülsing, **D. Joseph**, C. Majenz, and S. Yue

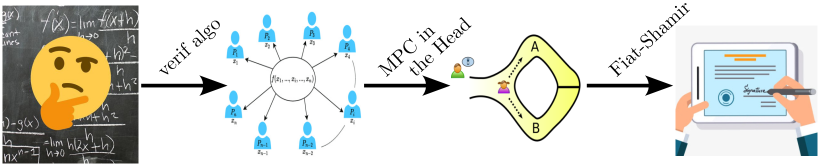
NIST Seminar - June 2023

Outline

1. **Hypercube MPC-in-the-Head:**
   How to make MPC-in-the-Head faster keeping the same proof size.
2. **Hypercube SDitH:**
   A smaller post-quantum signature based on Syndrome Decoding in the Head.
3. **Hypercube SDitH in the QROM:**
   Proof techniques for multi-round Fiat-Shamir transformed MPCitH schemes

Part I - Hypercube MPC-in-the-Head

Making digital signatures smaller and more secure

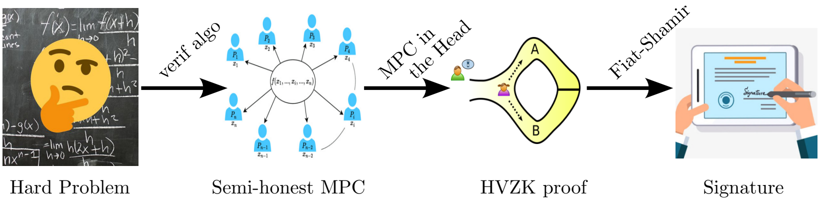


Hard Problem — Semi-honest MPC — HVZK proof — Signature

MPC-in-the-head + Fiat-Shamir

- **Hard instance:** Pick an instance of your favorite hard NP problem.
- **fast MPC:** Evaluate its verification function in MPC
- **MPC-in-the-head:** Turns it into a zero knowledge proof of knowledge – malicious prover
- **Fiat-Shamir:** make it non interactive and turns it in a strong digital signature
  - Security is the one of solving the hard NP problem.
  - Signing oracle access does not bring any advantage.

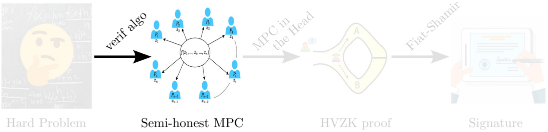
Making digital signatures smaller and more secure



MPC-in-the-head + Fiat-Shamir

- **Hard instance:** Pick an instance of your favorite hard NP problem.
- **fast MPC:** Evaluate its verification function in MPC
- **MPC-in-the-head:** Turns it into a zero knowledge proof of knowledge – malicious prover
- **Fiat-Shamir:** make it non interactive and turns it in a strong digital signature
  - Security is the one of solving the hard NP problem.
  - Signing oracle access does not bring any advantage.

Choice of MPC framework and algorithms



Picking an MPC framework

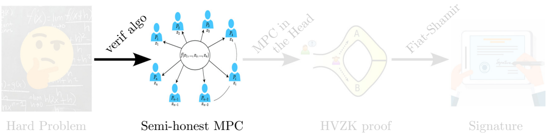
- Any number of players, the more, the better!
- Prefer linear/additive secret sharing protocol with public broadcasts.
- Target semi-honest security at this step
  malicious security is regained later
- Even a Trusted Dealer setup is ok!
  provide any triplets as part of the inputs, and make sure the algorithm checks the triplet consistency.

$\Longrightarrow$ MPCitH operates in the fastest and most concise out of all MPC settings

MPC algorithm: coding guidelines

- Optimize: |**inputs**| and |**communications**|, bonus: running time and rounds.

Choice of MPC framework and algorithms



Picking an MPC framework

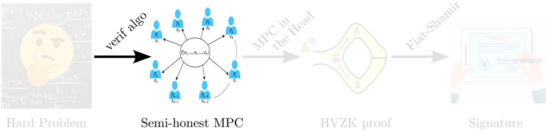
- Any number of players, the more, the better!
- Prefer linear/additive secret sharing protocol with public broadcasts.
- Target semi-honest security at this step
  malicious security is regained later
- Even a Trusted Dealer setup is ok!
  provide any triplets as part of the inputs, and make sure the algorithm checks the triplet consistency.

$\Longrightarrow$ MPCitH operates in the fastest and most concise out of all MPC settings

MPC algorithm: coding guidelines

- Optimize: |**inputs**| and |**communications**|, bonus: running time and rounds.

Choice of MPC framework and algorithms



Picking an MPC framework

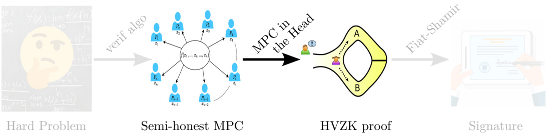
- Any number of players, the more, the better!
- Prefer linear/additive secret sharing protocol with public broadcasts.
- Target semi-honest security at this step
  malicious security is regained later
- Even a Trusted Dealer setup is ok!
  provide any triplets as part of the inputs, and make sure the algorithm checks the triplet consistency.

$\Longrightarrow$ MPCitH operates in the fastest and most concise out of all MPC settings

MPC algorithm: coding guidelines

- Optimize: |**inputs**| and |**communications**|, bonus: running time and rounds.

How MPC-in-the-Head works - Full Threshold security



Prover - Simulates the MPC protocol in the head

- Commits to everything that is secret (i.e. input secret-shares)
- Publishes everything that is public (i.e. broadcasted communications).

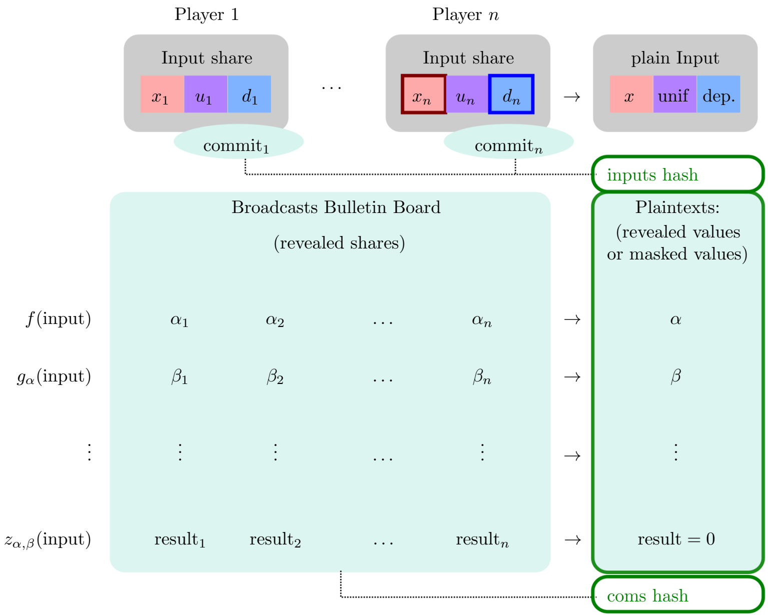
Verifier - checks the result and detects cheats

- Asks the prover to open $N-1$ parties inputs.
- Re-evaluate those parties and verify they have not cheated.

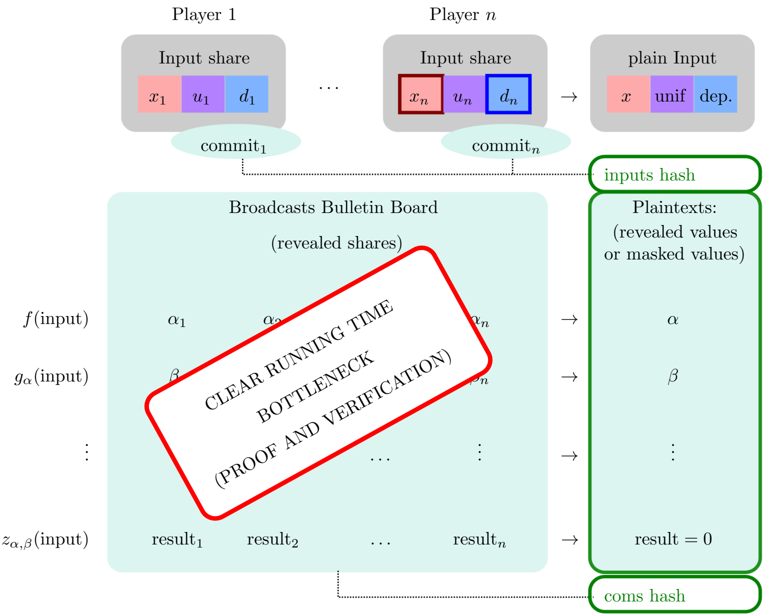
Bottom line: HVZK proof

- The verifier does not learn anything except the result.
- A prover that commits to secret shares that do not pass the verification function, gets caught with proba $1-\frac{1}{N}$

Complexity of MPC-in-the-Head



Complexity of MPC-in-the-Head



Complexity of MPC-in-the-Head

Computing the Broadcasts Bulletin Board

- **Before:** n evaluations of the MPC protocol (bottleneck)
- **Hypercube-MPCitH:** $\log_2(n)$ evaluations of the MPC protocol (negligible)

Main idea

- **Before:** we evaluate each individual parties
- **Hypercube-MPCitH:**
  - We group parties together and evaluate only $\log_2(n)$ subsets of parties.
  - Groups of parties are defined geometrically by their coordinates on a Hypercube.

Partitioning the parties - Sub-MPC protocols

Party 1	Party 2	Party 3
x_1	$x2$	$x3$
Party 4	Party 5	Party 6
$x4$	$x5$	$x6$

Original 6-players Protocol (chances of cheating: 1/6):

Party 1: x_1	bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
Party 2: x_2	bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$
Party 3: x_3	bcasts: $\alpha_3, \beta_3, \ldots, \text{result}_3$
Party 4: x_4	bcasts: $\alpha_4, \beta_4, \ldots, \text{result}_4$
Party 5: x_5	bcasts: $\alpha_5, \beta_5, \ldots, \text{result}_5$
Party 6: x_6	bcasts: $\alpha_6, \beta_6, \ldots, \text{result}_6$

Partitioning the parties - Sub-MPC protocols

Party 1	Party 2	Party 3
x_1	$x2$	$x3$
Party 4	Party 5	Party 6
$x4$	$x5$	$x6$

Plaintext Protocol:

Plaintext: $x_1 + \cdots + x_6$ — plain bcasts: $\alpha, \beta, \ldots, \text{result}$

Original 6-players Protocol (chances of cheating: 1/6):

Party 1: x_1 — bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
Party 2: x_2 — bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$
Party 3: x_3 — bcasts: $\alpha_3, \beta_3, \ldots, \text{result}_3$
Party 4: x_4 — bcasts: $\alpha_4, \beta_4, \ldots, \text{result}_4$
Party 5: x_5 — bcasts: $\alpha_5, \beta_5, \ldots, \text{result}_5$
~~Party 6: x_6 — bcasts: $\alpha_6, \beta_6, \ldots, \text{result}_6$~~

Partitioning the parties - Sub-MPC protocols

Party 1	Party 2	Party 3
x_1	$x2$	$x3$

Party 4	Party 5	Party 6
$x4$	$x5$	$x6$

Plaintext Protocol:

Plaintext: $x_1 + \cdots + x_6$ plain bcasts: $\alpha, \beta, \ldots, \text{result}$

Original 6-players Protocol (chances of cheating: 1/6):

Party 1: x_1 bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
Party 2: x_2 bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$
Party 3: x_3 bcasts: $\alpha_3, \beta_3, \ldots, \text{result}_3$
Party 4: x_4 bcasts: $\alpha_4, \beta_4, \ldots, \text{result}_4$
Party 5: x_5 bcasts: $\alpha_5, \beta_5, \ldots, \text{result}_5$
~~Party 6: x_6 bcasts: $\alpha_6, \beta_6, \ldots, \text{result}_6$~~

Red Sub Protocol (chances of cheating: 1/2):

Group 1: $x_1 + x_2 + x_3$ bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
~~Group 2: $x_4 + x_5 + x_6$ bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$~~

Partitioning the parties - Sub-MPC protocols

SANDBOXAQ™

Party 1 x_1	Party 2 $x2$	Party 3 $x3$
Party 4 $x4$	Party 5 $x5$	Party 6 $x6$

Plaintext Protocol:

Plaintext: $x_1 + \cdots + x_6$ — plain bcasts: $\alpha, \beta, \ldots,$ result

Original 6-players Protocol (chances of cheating: 1/6):

Party 1: x_1	bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
Party 2: x_2	bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$
Party 3: x_3	bcasts: $\alpha_3, \beta_3, \ldots, \text{result}_3$
Party 4: x_4	bcasts: $\alpha_4, \beta_4, \ldots, \text{result}_4$
Party 5: x_5	bcasts: $\alpha_5, \beta_5, \ldots, \text{result}_5$
~~Party 6: x_6~~	~~bcasts: $\alpha_6, \beta_6, \ldots, \text{result}_6$~~

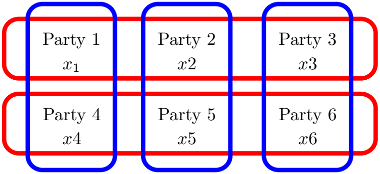
Red Sub Protocol (chances of cheating: 1/2):

Group 1: $x_1 + x_2 + x_3$	bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
~~Group 2: $x_4 + x_5 + x_6$~~	~~bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$~~

Blue Sub Protocol (chances of cheating: 1/3):

Group 1: $x_1 + x_4$	bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
Group 2: $x_2 + x_5$	bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$
~~Group 3: $x_3 + x_6$~~	~~bcasts: $\alpha_3, \beta_3, \ldots, \text{result}_3$~~

Partitioning the parties - Sub-MPC protocols



Plaintext Protocol:

Plaintext: $x_1 + \cdots + x_6$	plain bcasts: $\alpha, \beta, \ldots$, result

Original 6-players Protocol (chances of cheating: 1/6):

Party 1: x_1	bcasts: $\alpha_1, \beta_1, \ldots, \mathrm{result}_1$
Party 2: x_2	bcasts: $\alpha_2, \beta_2, \ldots, \mathrm{result}_2$
Party 3: x_3	bcasts: $\alpha_3, \beta_3, \ldots, \mathrm{result}_3$
Party 4: x_4	bcasts: $\alpha_4, \beta_4, \ldots, \mathrm{result}_4$
Party 5: x_5	bcasts: $\alpha_5, \beta_5, \ldots, \mathrm{result}_5$
~~Party 6: x_6~~	~~bcasts: $\alpha_6, \beta_6, \ldots, \mathrm{result}_6$~~

independent!!

Red Sub Protocol (chances of cheating: 1/2):

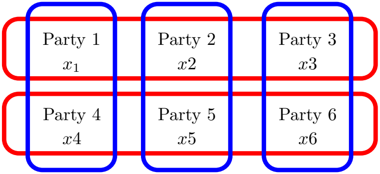
Group 1: $x_1 + x_2 + x_3$	bcasts: $\alpha_1, \beta_1, \ldots, \mathrm{result}_1$
~~Group 2: $x_4 + x_5 + x_6$~~	~~bcasts: $\alpha_2, \beta_2, \ldots, \mathrm{result}_2$~~

Blue Sub Protocol (chances of cheating: 1/3):

Group 1: $x_1 + x_4$	bcasts: $\alpha_1, \beta_1, \ldots, \mathrm{result}_1$
Group 2: $x_2 + x_5$	bcasts: $\alpha_2, \beta_2, \ldots, \mathrm{result}_2$
~~Group 3: $x_3 + x_6$~~	~~bcasts: $\alpha_3, \beta_3, \ldots, \mathrm{result}_3$~~

Partitioning the parties - Sub-MPC protocols

SANDBOXAQ™



Plaintext Protocol:

Plaintext: $x_1 + \cdots + x_6$ plain bcasts: $\alpha, \beta, \ldots, \mathrm{result}$

Original 6-players Protocol (chances of cheating: 1/6):

- Party 1: x_1 — bcasts: $\alpha_1, \beta_1, \ldots, \mathrm{result}_1$
- Party 2: x_2 — bcasts: $\alpha_2, \beta_2, \ldots, \mathrm{result}_2$
- Party 3: x_3 — bcasts: $\alpha_3, \beta_3, \ldots, \mathrm{result}_3$
- Party 4: x_4 — bcasts: $\alpha_4, \beta_4, \ldots, \mathrm{result}_4$
- Party 5: x_5 — bcasts: $\alpha_5, \beta_5, \ldots, \mathrm{result}_5$
- ~~Party 6: x_6 — bcasts: $\alpha_6, \beta_6, \ldots, \mathrm{result}_6$~~

1+5 evals

independent!!

Red Sub Protocol (chances of cheating: 1/2):

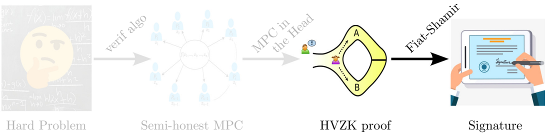
- Group 1: $x_1 + x_2 + x_3$ — bcasts: $\alpha_1, \beta_1, \ldots, \mathrm{result}_1$
- ~~Group 2: $x_4 + x_5 + x_6$ — bcasts: $\alpha_2, \beta_2, \ldots, \mathrm{result}_2$~~

Blue Sub Protocol (chances of cheating: 1/3):

- Group 1: $x_1 + x_4$ — bcasts: $\alpha_1, \beta_1, \ldots, \mathrm{result}_1$
- Group 2: $x_2 + x_5$ — bcasts: $\alpha_2, \beta_2, \ldots, \mathrm{result}_2$
- ~~Group 3: $x_3 + x_6$ — bcasts: $\alpha_3, \beta_3, \ldots, \mathrm{result}_3$~~

1+1+2 evals

Faster and Smaller proofs: pushing the tradeoff



Single MPC-in-the-head instance: $\log_2(n)$ bits of security

- Faster MPC-in-the-head that preserve soundness and small proof size
- Within the previous running time, we can take n larger

Parallel composition to achieve λ bits of security

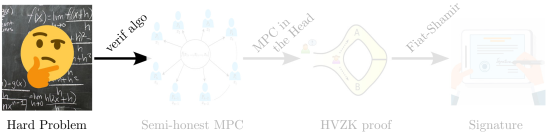
- Less parallel repetitions to achieve $1/2^\lambda$ security $\Longrightarrow$ smaller and faster.

Fiat-Shamir Transform

- HVZK proof with small communications $\Longrightarrow$ Small signature.

Part II - Hypercube SD-in-the-Head

The SD problem



The inhomogeneous SD problem

Given $H = (\mathrm{Id}_{m-k}||H')$ a random $m \times m-k$ matrix over $\mathbb{F}_q$, and a random syndrom $y \in \mathbb{F}_q^{m-k}$, find a solution $x \in \mathbb{F}_q^m$ of:

$$Hx = y \text{ where hamming weight}(x) \leq w$$

SD Verification in MPC (from [FJR22] at Crypto'22)

Equivalent formulation of the ISD problem (from [FJR22] at Crypto'22)

Given H' and y, find one vector $x_A \in \mathbb{F}_q^k$ and one polynomials $Q \in \mathbb{F}_q[X]$ monic of degree w and $P(X)$ of degree $\leq w-1$ such that

$$Q \times \mathtt{interpolation}_{[1,m]}(\underbrace{x_A||(y - H'x_A)}_{x}) - \underbrace{P \times (X-1)...(X-m)}_{\text{something zero over } [1,m]} = 0$$

Randomized verification function (w. false positive proba p)

Evaluate the above polynomial in MPC over just one random verifier-supplied point (in an extension field if needed). If the result is zero, the proof is accepted.

$$\text{Soundness of 1 iteration of SDitH: } (1-p)\left(1 - \frac{1}{N}\right)$$

SD Verification in MPC (from [FJR22] at Crypto'22)

Equivalent formulation of the ISD problem (from [FJR22] at Crypto'22)

Given H' and y, find one vector $x_A \in \mathbb{F}_q^k$ and one polynomials $Q \in \mathbb{F}_q[X]$ monic of degree w and $P(X)$ of degree $\leq w-1$ such that

$$Q \times \texttt{interpolation}_{[1,m]}(\underbrace{x_A || (y - H' x_A)}_{x}) - \underbrace{P \times (X-1)...(X-m)}_{\text{something zero over } [1,m]} = 0$$

Randomized verification function (w. false positive proba p)

Evaluate the above polynomial in MPC over just one random verifier-supplied point (in an extension field if needed). If the result is zero, the proof is accepted.

$$\text{Soundness of 1 iteration of SDitH: } (1-p)\left(1 - \frac{1}{N}\right)$$

Typical SD parameters

The SD and MPC parameters for our protocol, originally from [FJR22].

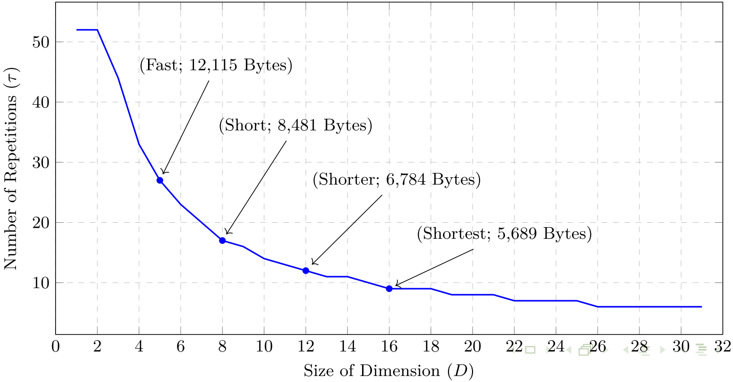
Scheme	SD Parameters					MPC Parameters			
	q	m	k	w	d	$\lvert\mathbb{F}_{\text{poly}}\rvert$	$\lvert\mathbb{F}_{\text{points}}\rvert$	t	p
Variant 1	2	1280	640	132	1	2^{11}	2^{22}	6	$\approx 2^{-69}$
Variant 2	2	1536	888	120	6	2^{8}	2^{24}	5	$\approx 2^{-79}$
Variant 3	2^8	256	128	80	1	2^{8}	2^{24}	5	$\approx 2^{-78}$

Signature sizes of SD-in-the-Head

SANDBOXAQ™

Our parameters with key and signature sizes in bytes for $\lambda = 128$.

Scheme	**Aim**	**Parameters**			**Sizes (in bytes)**		
		N	D	τ	pk	sk	Sign (Max)
Variant 3	Fast	2	5	27	144	16	12 115
	Short	2	8	17	144	16	8 481
	Shorter	2	12	12	144	16	6 784
	Shortest	2	16	9	144	16	5 689



Benchmarks and performance of Hypercube-SDitH

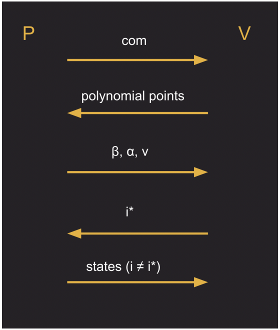
SANDBOXAQ™

Table 7: Reference implementation benchmarks of SDitH [FJR22] vs our scheme for $\lambda = 128$. Both ran on a single CPU core of a 3.1 GHz Intel Core i9-9990K.

Scheme	Aim	Signature Size	Parameters			Sign Time (in ms)			Verify Time
			N	D	τ	Offline	Online	Total	(in ms) Total
SDitH [FJR22] (Variant 3)	Fast	12 115	32	-	27	0.87	5.03	5.96	4.74
	Short	8 481	256	-	17	4.33	18.95	23.56	20.80
	Shorter	6 784	2^{12}	-	12	59.24	251.14	313.70	244.30
	Shortest	5 689	2^{16}	-	9	-	-	-	-
Ours (Variant 3)	Fast	12 115	2	5	27	0.47	0.83	1.30	0.98
	Short	8 481	2	8	17	2.26	0.61	2.87	2.59
	Shorter	6 784	2	12	12	25.93	0.50	26.43	25.79
	Shortest	5 689	2	16	9	320.24	0.42	320.66	312.67

Part III - Hypercube SDitH in the QROM

Security of the Fiat-Shamir transform in the QROM

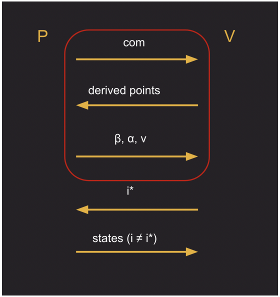


Five round protocol

- Most MPCitH signatures are presented as FS-transformed 5 or 7 round identity scheme
- Zero knowledge comes from the final challenge

Security of the Fiat-Shamir transform in the QROM

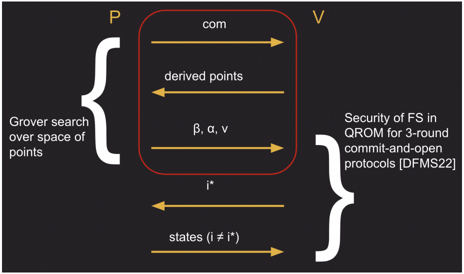
SANDBOXAQ™



Three round protocol

- Argument: one can view the protocol as a 3 round commit-and-open protocol
- View checking points as an internal derivation rather than external challenge
- Still requires parallel-composed derivation

Security of the Fiat-Shamir transform in the QROM



Grover plus [DFMS22]

- Adversary can Grover search over derived point space (first ‘challenge’)
- Can apply 3-round commit-and-open QROM security bound in the second ‘challenge’

Change versus 5 round

- Scheme is mechanically the same (parallel composed across all τ iterations)
- Optimal attack is still due to KZ split across both levels of ‘challenges’

Conclusion and perspectives

Part of this work was included in a post quantum signature candidate for NIST (To be presented separately)

- Multiple techniques: Hypercube-SDitH, Threshold-SDiTH
- Security analysis in the QROM model (vs. ROM)
- Parameters for $\lambda = 128, 192$ and 256
- SD over GF256 and over prime fields

Other goodies

- Offline/Online phase separation: Online phase with μs latency
- Can be applied to other hard problems.

Open problem / Limitation

- State generation is still in $O(n)$: we cannot take n exponential
- $\implies$ randomness generation becomes the bottleneck for Hypercube-SDiTH.

Thank you!