Hypercube SDiT: 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
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

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

**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.
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.*
  
  → 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.

⇒ 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.

\[ \Rightarrow \text{MPCitH operates in the fastest and most concise out of all MPC settings} \]

MPC algorithm: coding guidelines

- Optimize: \(|\text{inputs}|\) and \(|\text{communications}|\), 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 $1 - \frac{1}{N}$.
Hypercube MPC-in-the-Head

Complexity of MPC-in-the-Head

Player 1

Input share
\[ x_1 \quad u_1 \quad d_1 \]

commit_1

\[ \ldots \]

Player \( n \)

Input share
\[ x_n \quad u_n \quad d_n \]

commit_n

\[ \rightarrow \]

plain Input
\[ x \quad \text{unif dep.} \]

Broadcasts Bulletin Board
(revealed shares)

\[ f(\text{input}) \]
\[ g_\alpha(\text{input}) \]
\[ z_{\alpha,\beta}(\text{input}) \]

(result)

\[ \alpha_1 \quad \alpha_2 \quad \ldots \quad \alpha_n \]

\[ \beta_1 \quad \beta_2 \quad \ldots \quad \beta_n \]

\[ \text{result}_1 \quad \text{result}_2 \quad \ldots \quad \text{result}_n \]

\[ \rightarrow \]

\[ \alpha \]

\[ \beta \]

\[ : \quad : \quad : \quad : \quad : \]

\[ \rightarrow \]

\[ \text{result} = 0 \]

Plaintexts:
(revealed values or masked values)

inputs hash

\[ \rightarrow \]

coms hash
Complexity of MPC-in-the-Head

Player 1

Input share
\[ x_1 \quad u_1 \quad d_1 \]
commit \(_1\)

Player \(n\)

Input share
\[ x_n \quad u_n \quad d_n \]
commit \(_n\)

plain Input
\[ x \quad \text{unif dep.} \]

Broadcasts Bulletin Board
(revealed shares)

Plaintexts:
(revealed values or masked values)

\[ f(\text{input}) \]
\[ g_\alpha(\text{input}) \]
\[ z_{\alpha,\beta}(\text{input}) \]

\[ \alpha_1 \quad \alpha_2 \quad \ldots \quad \alpha_n \]
\[ \beta_1 \quad \beta_2 \quad \ldots \quad \beta_n \]

result\(_1\) result\(_2\) \ldots result\(_n\)

\[ \text{result} = 0 \]

CLEAR RUNNING TIME
PROOF AND VERIFICATION

inputs hash

coms hash
Hypercube 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

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

Party 1: $x_1$

Party 2: $x_2$

Party 3: $x_3$

Party 4: $x_4$

Party 5: $x_5$

Party 6: $x_6$

bcast: $\alpha_1, \beta_1, \ldots, \text{result}_1$

bcast: $\alpha_2, \beta_2, \ldots, \text{result}_2$

bcast: $\alpha_3, \beta_3, \ldots, \text{result}_3$

bcast: $\alpha_4, \beta_4, \ldots, \text{result}_4$

bcast: $\alpha_5, \beta_5, \ldots, \text{result}_5$

bcast: $\alpha_6, \beta_6, \ldots, \text{result}_6$
Partitioning the parties - Sub-MPC protocols

<table>
<thead>
<tr>
<th>Party 1</th>
<th>Party 2</th>
<th>Party 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
</tr>
<tr>
<td>Party 4</td>
<td>Party 5</td>
<td>Party 6</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$x_5$</td>
<td>$x_6$</td>
</tr>
</tbody>
</table>

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

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: $x_1$
- Party 2: $x_2$
- Party 3: $x_3$
- Party 4: $x_4$
- Party 5: $x_5$
- Party 6: $x_6$

  - bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
  - bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$
  - bcasts: $\alpha_3, \beta_3, \ldots, \text{result}_3$
  - bcasts: $\alpha_4, \beta_4, \ldots, \text{result}_4$
  - bcasts: $\alpha_5, \beta_5, \ldots, \text{result}_5$
  - bcasts: $\alpha_6, \beta_6, \ldots, \text{result}_6$

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

- Group 1: $x_1 + x_2 + x_3$
- Group 2: $x_4 + x_5 + x_6$

  - bcasts: $\alpha_1, \beta_1, \ldots, \text{result}_1$
  - bcasts: $\alpha_2, \beta_2, \ldots, \text{result}_2$
Partitioning the parties - Sub-MPC protocols

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: $\alpha_2, \beta_2, \ldots, \text{result}_2$

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: $\alpha, \beta, \ldots, \text{result}$

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: $\alpha_2, \beta_2, \ldots, \text{result}_2$

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: $\alpha, \beta, \ldots, \text{result}$

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

- Party 1: $x_1$
- Party 2: $x_2$
- Party 3: $x_3$
- Party 4: $x_4$
- Party 5: $x_5$
- Party 6: $x_6$

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

1+5 evals

- Group 1: $x_1 + x_2 + x_3$
- Group 2: $x_4 + x_5 + x_6$

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

1+1+2 evals

- Group 1: $x_1 + x_4$
- Group 2: $x_2 + x_5$
- Group 3: $x_3 + x_6$
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 $\lambda$ bits of security
- Less parallel repetitions to achieve $1/2^\lambda$ security $\Rightarrow$ smaller and faster.

Fiat-Shamir Transform
- HVZK proof with small communications $\Rightarrow$ Small signature.
Part II - Hypercube SD-in-the-Head
The inhomogeneous SD problem

Given $H = (\text{Id}_{m-k} \mid H')$ 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:

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

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

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.

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

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

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.

Soundness of 1 iteration of SDitH: $(1 - p)\left(1 - \frac{1}{N}\right)$
The SD and MPC parameters for our protocol, originally from [FJR22].

<table>
<thead>
<tr>
<th>Scheme</th>
<th>SD Parameters</th>
<th>MPC Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$q$</td>
<td>$m$</td>
</tr>
<tr>
<td>Variant 1</td>
<td>2</td>
<td>1280</td>
</tr>
<tr>
<td>Variant 2</td>
<td>2</td>
<td>1536</td>
</tr>
<tr>
<td>Variant 3</td>
<td>$2^8$</td>
<td>256</td>
</tr>
</tbody>
</table>
Signature sizes of SD-in-the-Head

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

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Aim</th>
<th>Parameters</th>
<th>Sizes (in bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$N$</td>
<td>$D$</td>
</tr>
<tr>
<td>Variant 3</td>
<td>Fast</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Short</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Shorter</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Shortest</td>
<td>2</td>
<td>16</td>
</tr>
</tbody>
</table>

![Graph showing the relationship between the size of dimension ($D$) and the number of repetitions ($\tau$).](https://example.com/graph.png)

(Fast; 12,115 Bytes)
(Short; 8,481 Bytes)
(Shorter; 6,784 Bytes)
(Shortest; 5,689 Bytes)
Benchmarks and performance of Hypercube-SDitH

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.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Aim</th>
<th>Signature Size</th>
<th>Parameters</th>
<th>Sign Time (in ms)</th>
<th>Verify Time (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$N$</td>
<td>$D$</td>
<td>$\tau$</td>
</tr>
<tr>
<td>SDitH [FJR22](Variant 3)</td>
<td>Fast</td>
<td>12 115</td>
<td>32</td>
<td>-</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Short</td>
<td>8 481</td>
<td>256</td>
<td>-</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Shorter</td>
<td>6 784</td>
<td>$2^{12}$</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Shortest</td>
<td>5 689</td>
<td>$2^{16}$</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>Ours (Variant 3)</td>
<td>Fast</td>
<td>12 115</td>
<td>2</td>
<td>5</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Short</td>
<td>8 481</td>
<td>2</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Shorter</td>
<td>6 784</td>
<td>2</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Shortest</td>
<td>5 689</td>
<td>2</td>
<td>16</td>
<td>9</td>
</tr>
</tbody>
</table>
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

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

- 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 $\tau$ 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 $\mu s$ latency
- Can be applied to other hard problems.

Open problem / Limitation

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