Zero-Knowledge Protocols for the Subset Sum Problem from MPC-in-the-Head with Rejection

Thibauld Feneuil[1,2] Jules Maire[3]
Matthieu Rivain[1] Damien Vergnaud[3,4]

1. CryptoExperts, Paris, France

2. Sorbonne Université, CNRS, INRIA, Institut de Mathématiques de Jussieu-Paris Rive Gauche, Ouragan, Paris, France

3. Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

4. Institut Universitaire de France

ASIACRYPT'22. *December 7, 2022.*

Zero-Knowledge Proofs for Subset Sum Problem

Subset Sum Problem

From (w, t), find a vector x such that

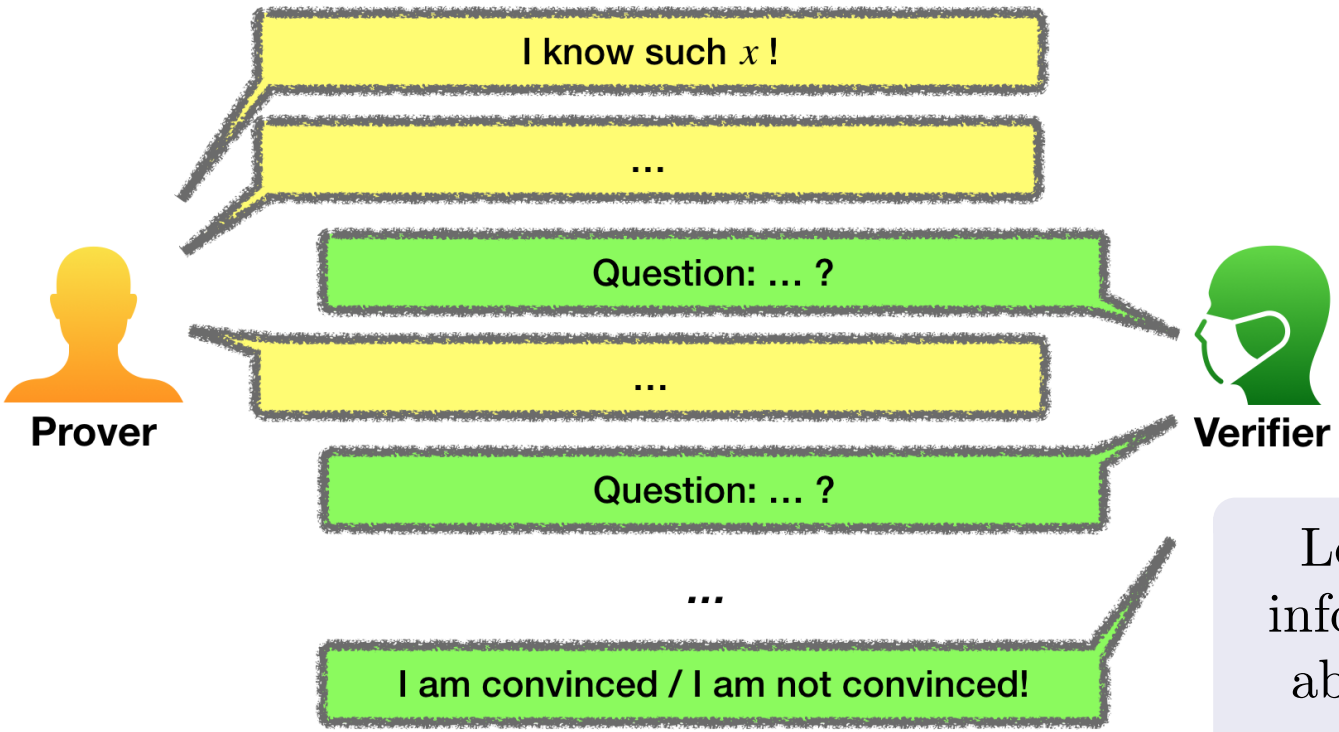
$$\langle w, x \rangle = t \mod q \quad \text{and} \quad x \in \{0,1\}^n.$$

Zero-Knowledge Proofs for Subset Sum Problem

Subset Sum Problem

From (w, t), find a vector x such that

$$\langle w, x \rangle = t \mod q \quad \text{and} \quad x \in \{0,1\}^n.$$



Learn no information about the secret x.

MPC-in-the-Head Paradigm

MPC-in-the-Head Paradigm

- Generic technique to build *zero-knowledge protocols* using *multi-party computation*.

- Introduced in 2007 by:

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. *Zero-knowledge from secure multiparty computation.* STOC 2007.

Sharing of the secret

The secret x satisfies

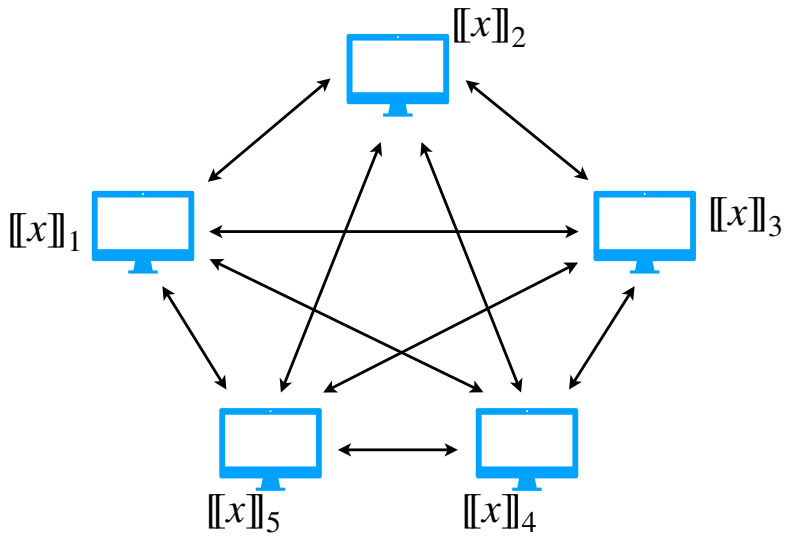
$$\langle w, x \rangle = t \mod q \quad \text{and} \quad x \in \{0,1\}^n.$$

We share it in N parts:

$$x = [\![x]\!]_1 + [\![x]\!]_2 + \ldots + [\![x]\!]_{N-1} + [\![x]\!]_N + \Delta x.$$

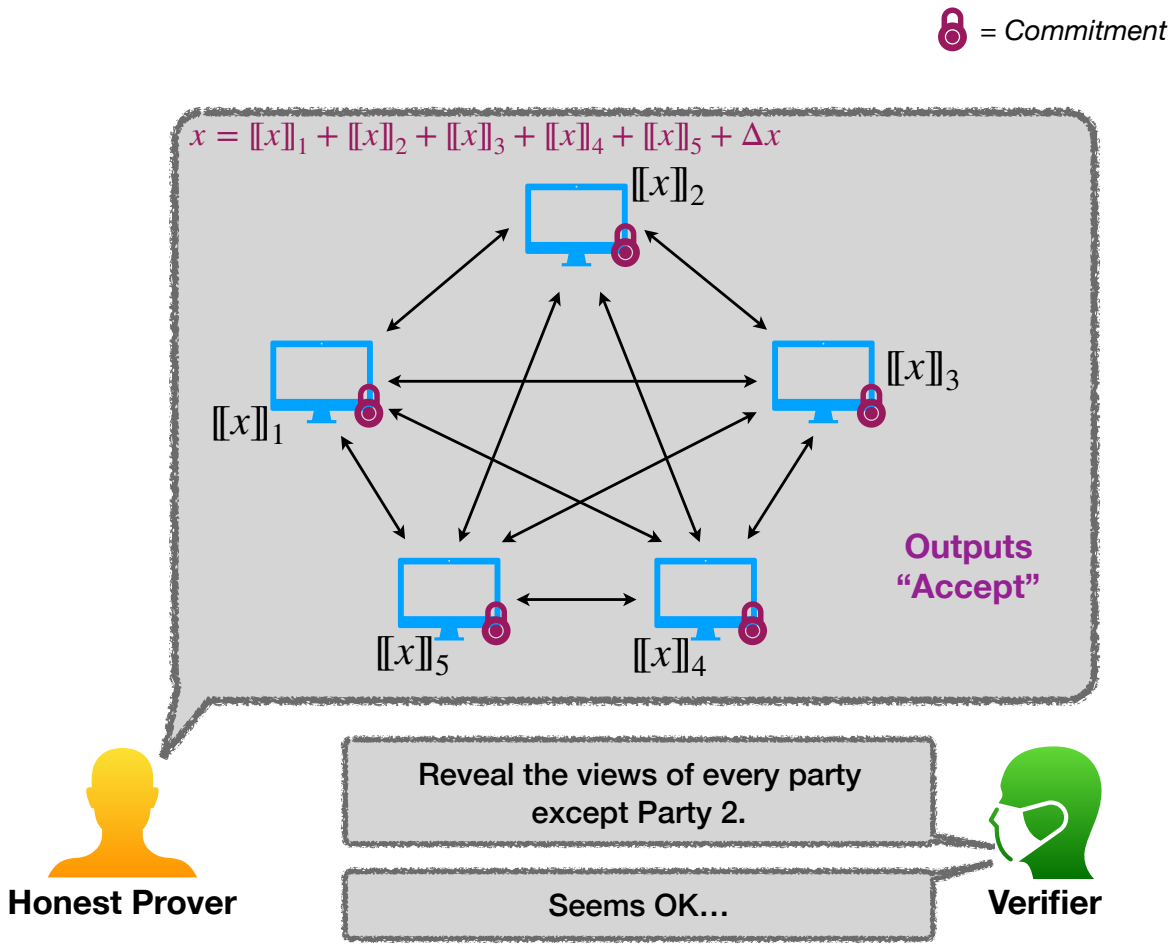
MPC-in-the-Head Paradigm

$$x = [\![x]\!]_1 + [\![x]\!]_2 + [\![x]\!]_3 + [\![x]\!]_4 + [\![x]\!]_5 + \Delta x$$

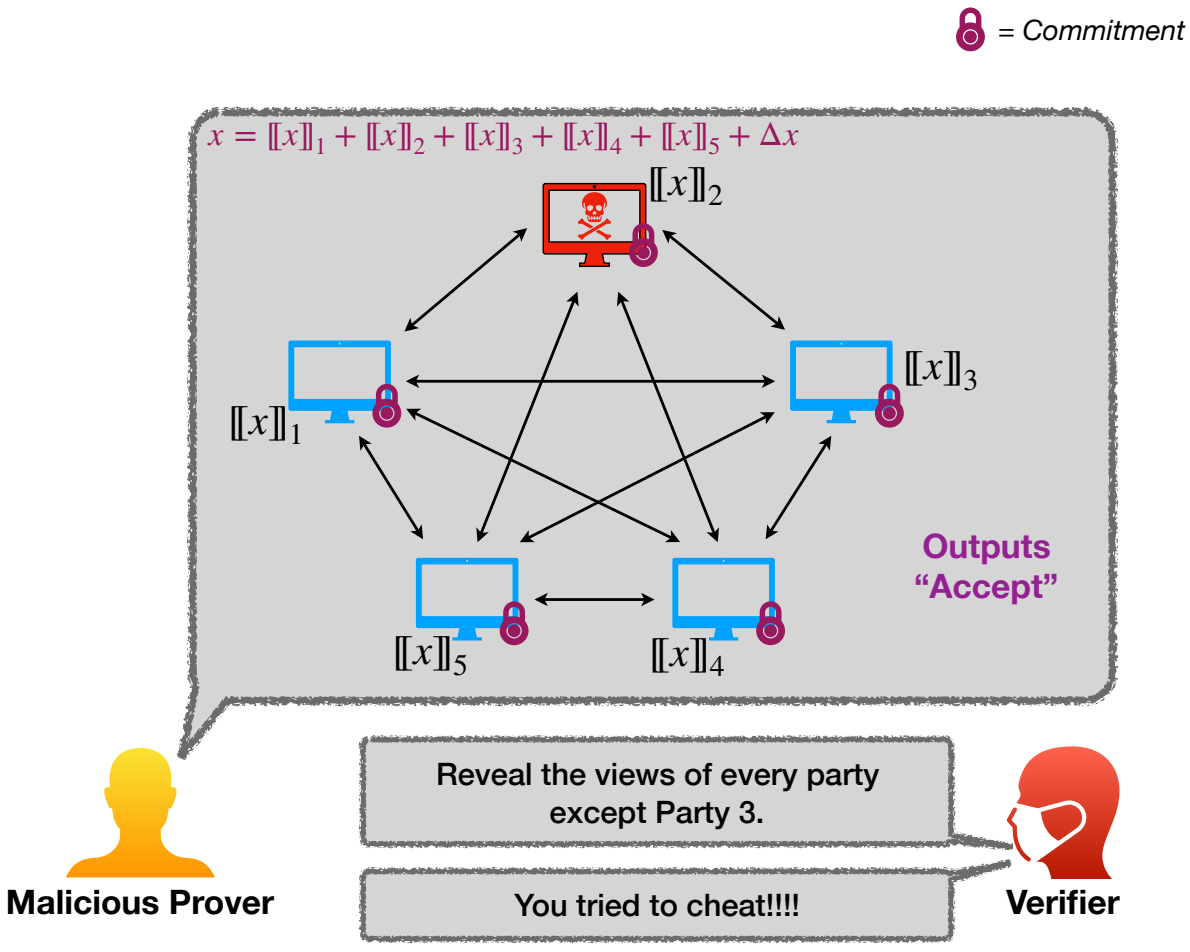


The multi-party computation outputs
- *Accept* if x is a subset sum solution,
- *Reject* otherwise.

MPC-in-the-Head Paradigm



MPC-in-the-Head Paradigm



MPC-in-the-Head Paradigm - Performances

Soundness error:

$$\frac{1}{N}$$

MPC-in-the-Head Paradigm - Performances

Soundness error:

$$\frac{1}{N}$$

Proof transcript:

- Inputs of $N-1$ parties:

	$\mathcal{P}_1$		$\mathcal{P}_2$		$\dots$		$\mathcal{P}_{N-1}$		$\mathcal{P}_N$		
x	$= [\![x]\!]_1$	$+$	$[\![x]\!]_2$	$+$	$\dots$	$+$	$[\![x]\!]_{N-1}$	$+$	$[\![x]\!]_N$	$+$	Δx
	$\uparrow$		$\uparrow$				$\uparrow$		$\uparrow$		
	seed_1		seed_2				seed_{N-1}		seed_N		

MPC-in-the-Head Paradigm - Performances

Soundness error:

$$\frac{1}{N}$$

Proof transcript:

- Inputs of $N-1$ parties:
  - $N-1$ seeds of λ bits
  - Sharing offset: Δx
- ...

MPC-in-the-Head Paradigm - Performances

Soundness error:

$$\frac{1}{N}$$

Proof transcript:

- Inputs of $N-1$ parties:
  - $N-1$ seeds of λ bits
  - Sharing offset: Δx
- ...

To achieve the soundness error of $2^{-\lambda}$, one needs to repeat the protocol

$$\tau := \frac{\lambda}{\log_2 N} \text{ times.}$$

How to share a bit

How to share a bit $b \in \{0,1\}$ such that

$$b = [\![b]\!]_1 + [\![b]\!]_2 + \ldots + [\![b]\!]_N + \Delta b \ ?$$

How to share a bit

How to share a bit $b \in \{0,1\}$ such that

$$b = [\![b]\!]_1 + [\![b]\!]_2 + \ldots + [\![b]\!]_N + \Delta b \ ?$$

☞ Additive sharing (modulo q)

$$\begin{cases} [\![b]\!]_i & \xleftarrow{\$} \mathbb{Z}_q \text{ for all } i \in [N], \\ \Delta b & \leftarrow b - \sum_{i=1}^{N} [\![b]\!]_i \bmod q. \end{cases}$$

How to share a bit

How to share a bit $b \in \{0,1\}$ such that

$$b = [\![b]\!]_1 + [\![b]\!]_2 + \ldots + [\![b]\!]_N + \Delta b \ ?$$

☞ Additive sharing (modulo q)

$$\begin{cases} [\![b]\!]_i & \xleftarrow{\$} \mathbb{Z}_q \text{ for all } i \in [N], \\ \Delta b & \leftarrow b - \sum_{i=1}^{N} [\![b]\!]_i \bmod q. \end{cases}$$

$$\begin{aligned} \textsc{Size}_{\text{Proof}} &\geq \tau \cdot \textsc{Size}_{\Delta x} \\ &= \tau \cdot (n \log_2 q) = 128 \text{ KB} \end{aligned}$$

$n = 256,\ q = 2^{256}$

$N = 256,\ \lambda = 128,\ \tau = 16$

How to share a bit

How to share a bit $b \in \{0,1\}$ such that

$$b = [\![b]\!]_1 + [\![b]\!]_2 + \ldots + [\![b]\!]_N + \Delta b \ ?$$

☞ Additive sharing (modulo q)

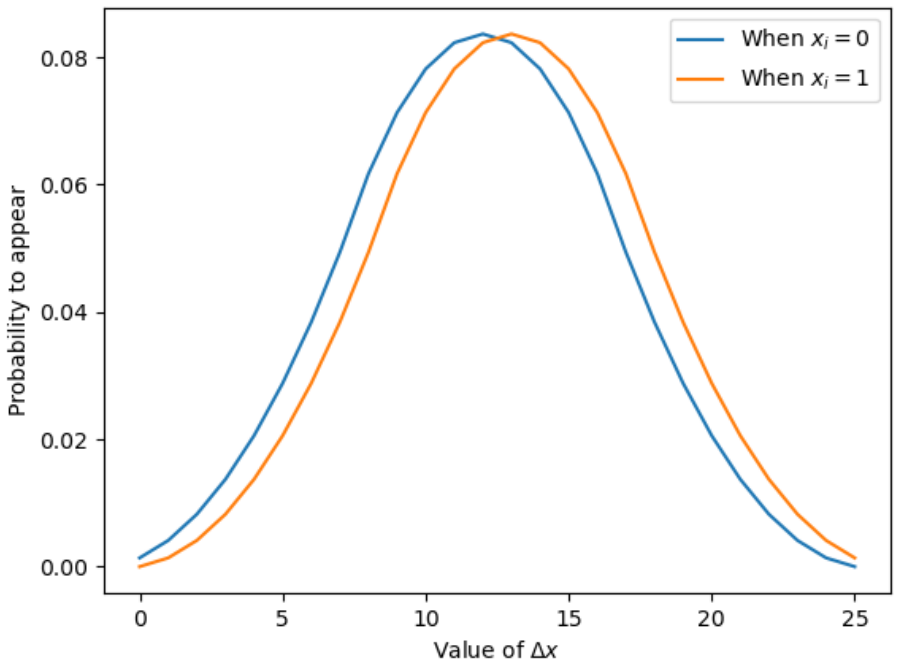
$$\begin{cases} [\![b]\!]_i & \xleftarrow{\$} \mathbb{Z}_q \text{ for all } i \in [N], \\ \Delta b & \leftarrow b - \sum_{i=1}^{N} [\![b]\!]_i \bmod q. \end{cases}$$

☞ Additive sharing on integers

$$\begin{cases} [\![b]\!]_i & \xleftarrow{\$} \{0, \ldots, A-1\} \text{ for all } i \in [N], \\ \Delta b & \leftarrow b - \sum_{i=1}^{N} [\![b]\!]_i. \end{cases}$$

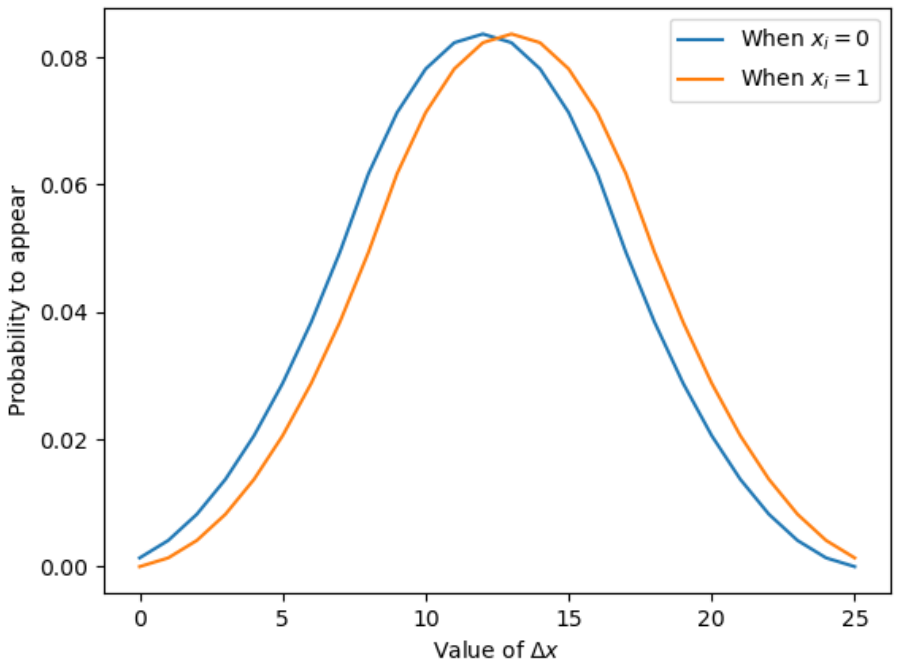
Sharing on integers - Information leakage

The sharing offset Δb leaks information.



Sharing on integers - Information leakage

The sharing offset Δb leaks information.

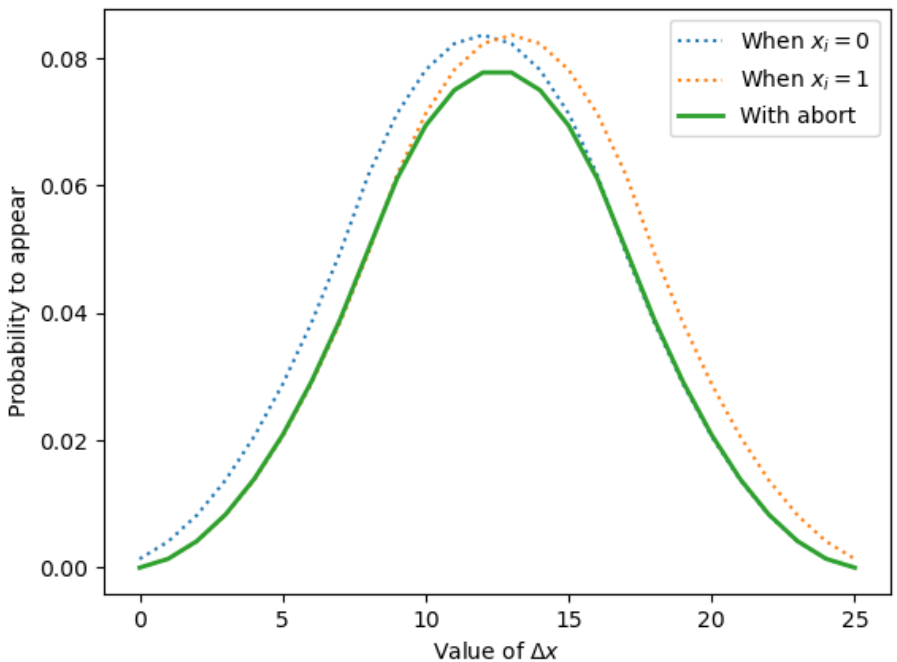


Solution: add a rejection rule.

Sharing on integers - Rejection Rule

The prover must abort when

- we have $b = 0$ and $[\![b]\!]_{i^*} = A - 1$,
- we have $b = 1$ and $[\![b]\!]_{i^*} = 0$,



Sharing on integers - Rejection Rule

The prover must abort when

- we have $b = 0$ and $[\![b]\!]_{i^*} = A - 1$,
- we have $b = 1$ and $[\![b]\!]_{i^*} = 0$,

Rejection rate:

$$\frac{1}{A}$$

Sharing offset cost:

$$\log_2(A-1) \text{ bits}$$

instead of $\log_2 q$ bits.

Sharing on integers - Rejection Rule

The prover must abort when

- we have $b = 0$ and $[\![b]\!]_{i^*} = A - 1$,
- we have $b = 1$ and $[\![b]\!]_{i^*} = 0$,

Rejection rate:

$$\frac{1}{A}$$

Aborting leaks no information about the secret.

Sharing offset cost:

$$\log_2(A-1) \text{ bits}$$

instead of $\log_2 q$ bits.

Sharing on integers - Binary vector

We can generalize to a binary vector $x \in \{0,1\}^n$.

Sharing on integers - Binary vector

We can generalize to a binary vector $x \in \{0,1\}^n$.

Rejection rate:

$$1 - \left(1 - \frac{1}{A}\right)^n$$

Sharing offset cost:

$$n \cdot \log_2(A-1) \text{ bits}$$

instead of $n \cdot \log_2 q$ bits.

Sharing on integers - Binary vector

We can generalize to a binary vector $x \in \{0,1\}^n$.

Rejection rate:

$$1 - \left(1 - \frac{1}{A}\right)^n$$

Sharing offset cost:

$$n \cdot \log_2(A-1) \text{ bits}$$

instead of $n \cdot \log_2 q$ bits.

128 KB

Sharing on integers - Binary vector

We can generalize to a binary vector $x \in \{0,1\}^n$.

Rejection rate:

$$1 - \left(1 - \frac{1}{A}\right)^n$$

Rejection: 22%

Sharing offset cost:

7 KB

$$n \cdot \log_2(A-1) \text{ bits}$$

instead of $n \cdot \log_2 q$ bits.

128 KB

MPC protocol - Subset Sum Problem

The multi-party computation must check that the vector x satisfies

$$\underbrace{\langle w, x \rangle = t}_{\text{linear, easy to check}} \quad \text{and} \quad \underbrace{x \in \{0,1\}^n}_{\text{non-linear}}$$

To check that $x \in \{0,1\}^n$:

- Using batch product verification [BN20],
- Using cut-and-choose strategy [KKW18].

MPC protocol - Batch Product Verification

Strategy 1:

To check that $x \in \{0,1\}^n$, we will check using *standard techniques* ([BN20]) that

$$x \circ (x - 1) = 0 \mod q'$$

where

- $\circ$ is the pointwise multiplication, and
- q' is a prime close to A.

[BN20] Carsten Baum and Ariel Nof. *Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography.* PKC 2020.

MPC protocol - Cut-and-choose strategy

Strategy 2:

To check that $x \in \{0,1\}^n$, we will

- get $[\![r]\!]$ where r is a **random mask which is ensured to be binary by a cut-and-choose phase [KKW18].**
- compute and reveal $\tilde{x}$ as $x \oplus r \in \{0,1\}^n$,
- deduce a sharing of x using the relation

$$[\![x]\!] = (1 - \tilde{x}) \circ [\![r]\!] + \tilde{x} \circ (1 - [\![r]\!]).$$

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. *Improved non-interactive zero knowledge with applications to post-quantum signatures.* ACM CCS 2018.

Decreasing the Rejection Rate

The verifier validates the proof transcript when the prover does not abort among the τ repetitions.

Decreasing the Rejection Rate

~~The verifier validates the proof transcript when the prover does not abort among the τ repetitions.~~

The verifier validates the proof transcript when the prover aborts at most η times among the τ repetitions.

- ☞ Decrease a lot the rejection rate
- ☞ Increase a bit the soundness error

Performances

Interactive proof of knowledge for the subset sum problem.

Protocol	Parameters					Proof size	Rej. rate
	τ	η	N	A	M		
Shamir [Sha86]	219	-	-	-	-	1186 KB	-
[LNSW13]	219	-	-	-	-	2350 KB	-
Beullens [Beu20]	14	-	1024	-	4040	122 KB	-
Batching Strategy	17	0	256	2^{13}	-	16.6 KB	0.412
Batching Strategy	21	3	256	2^{13}	-	17.7 KB	0.004
C&C Strategy	19	0	256	2^{13}	954	13.0 KB	0.448
C&C Strategy	24	3	256	2^{14}	952	15.4 KB	0.001

with $n = 256$ and $q = 2^{256}$ as parameters for the subset sum problem.

ZK PoK for SIS Problem

Given (A, u), prove the knowledge of a vector s such that

$$As = u \mod q \quad \text{and} \quad \|s\|_\infty \leq \beta.$$

ZK PoK for SIS Problem

Given (A, u), prove the knowledge of a vector s such that

$$As = u \mod q \quad \text{and} \quad \|s\|_\infty \leq \beta.$$

Protocol	Any q	Instance 1		Instance 2	
		Proof Size	Rej. Rate	Proof Size	Rej. Rate
[LNSW13]	✓	3600 KB	-	8988 KB	-
[BN20]	q prime	-	-	4077 KB	-
[Beu20]	q prime	233 KB	-	444 KB	-
Batching Strategy	✓	291 KB	0.04	291 KB	0.04
C&C Strategy	✓	184 KB	0.05	184 KB	0.05
[BLS19]	q prime + NTT	384 KB	0.92	$q \approx 2^{61}$	
[ENS20]	q prime + NTT	47 KB	0.95		
[LNS21]	q prime + NTT	33.3 KB	0.85		
Aurora	q prime + NTT	71 KB	-		
Ligero	q prime + NTT	157 KB	-		

$q \approx 2^{32}$

ZK PoK for secret keys in TFHE

In TFHE, prove the knowledge of key-plaintext pair matching a given ciphertext ($q = 2^{64}$).

Protocol	Parameters					Proof size	Rej. rate
	τ	η	N	A	M		
1 ciphertext							
Shamir [Sha86]	219	-	-	-	-	845 KB	-
Batching Strategy	19	2	256	2^{15}	-	46.1 KB	0.007
C&C Strategy	24	3	256	2^{15}	952	34.0 KB	0.002
1024 ciphertexts							
Shamir [Sha86]	219	-	-	-	-	77.9 MB	-
Batching Strategy	19	2	256	2^{22}	-	5.90 MB	0.003
C&C Strategy	24	3	256	2^{21}	952	3.65 MB	0.006

Signature Scheme from BHH01 PRF

Boneh-Halevi-Howgrave-Graham PRF:

$$F_x(m) = \mathrm{MSB}_\delta((x+m)^{-1} \bmod p)$$

Signature:

- Private key: $x \in \mathbb{Z}_p$,
- Public key: $y_1 := F_x(1), \ldots, y_t := F_x(t)$

Proof of knowledge of x, z_1, ..., z_t such that

$$\text{for all } i, \quad (x+i)(2^{(1-\delta)m} y_i + z_i) \equiv 1 \bmod p$$

$$\text{and} \quad z_1, \ldots, z_t \in \{0, \ldots, 2^{(1-\delta)m} - 1\}$$

Signature Scheme from BHH01 PRF

Boneh-Halevi-Howgrave-Graham PRF:

$$F_x(m) = \mathrm{MSB}_\delta((x+m)^{-1} \bmod p)$$

Signature:

- Private key: $x \in \mathbb{Z}_p$,
- Public key: $y_1 := F_x(1), \ldots, y_t := F_x(t)$

Parameters					Size	p_{rej}
$p \approx 2^m$	t	δ	N	τ		
$\approx 2^{229}$	3	88/229	256	16	4 916 B	0.012
$\approx 2^{186}$	4	58/186	256	16	4 860 B	0.016
$\approx 2^{175}$	5	47/175	256	16	5 074 B	0.019

Conclusion

Summary

- ☞ New sharing method when large modulus
- ☞ Rejection rule to avoid information leakage
- ☞ Efficient ZK PoK for the subset sum problem
- ☞ Interesting applications (FHE, signature, ...)

Perspective

- ☞ Other applications
- ☞ Quantum cryptanalysis for Boneh *et al.*'s PRF

More details in `https://eprint.iacr.org/2022/223`.