Non-Interactive Secure Computation of Inner-Product from LPN and LWE

Geoffroy Couteau, Maryam Zarezadeh
Non-Interactive Key Exchange

A very appealing interaction pattern:
- \( n \) parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the \( \Omega(n^2) \) overhead of naive pairwise exchange

Non-Interactive Secure Computation

This Work
Non-Interactive Key Exchange

A very appealing interaction pattern:
• $n$ parties simultaneously broadcast a single message
• All pairs of parties get a shared private key
• Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

This Work

$K_{14} \leftarrow \text{Key}(sk_1, pk_4)$

$K_{14} \leftarrow \text{Key}(sk_4, pk_1)$
A very appealing interaction pattern:
• $n$ parties simultaneously broadcast a single message
• All pairs of parties get a shared private key
• Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange
A very appealing interaction pattern:
- $n$ parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange
Non-Interactive Key Exchange

A very appealing interaction pattern:
• $n$ parties simultaneously broadcast a single message
• All pairs of parties get a shared private key
• Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

This Work
Non-Interactive Key Exchange

A very appealing interaction pattern:
- $n$ parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

Can we get a similar pattern for some simple MPC?
- $n$ parties broadcast an *encoding* of their input
- Pairs $(P_i, P_j)$ can compute $f_i(x_i, x_j)$ and $f_j(x_i, x_j)$ from their state and the other party’s encoding
- Avoids the $\Omega(n^2)$ overhead

This Work
Non-Interactive Key Exchange

A very appealing interaction pattern:
• $n$ parties simultaneously broadcast a single message
• All pairs of parties get a shared private key
• Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

Non-Interactive Secure Computation

Can we get a similar pattern for some simple MPC?
• $n$ parties broadcast an encoding of their input
• Pairs $(P_i, P_j)$ can compute $f_i(x_i, x_j)$ and $f_j(x_i, x_j)$ from their state and the other party’s encoding
• Avoids the $\Omega(n^2)$ overhead

This Work

\[
f_1(x_1, x_4) = \text{Out}(\text{st}_1, \text{Enc}(x_4))
\]

\[
f_1(x_1, x_4) = \text{Out}(\text{st}_4, \text{Enc}(x_1))
\]
**Non-Interactive Key Exchange**

A very appealing interaction pattern:

- $n$ parties simultaneously broadcast a single message
- All pairs of parties get a shared private key
- Avoids the $\Omega(n^2)$ overhead of naive pairwise exchange

**Non-Interactive Secure Computation**

Can we get a similar pattern for some simple MPC?

- $n$ parties broadcast an encoding of their input
- Pairs $(P_i, P_j)$ can compute $f_i(x_i, x_j)$ and $f_j(x_i, x_j)$ from their state and the other party’s encoding
- Avoids the $\Omega(n^2)$ overhead

**This Work**

- Non-interactive MPC for shares of inner products: $f_i(x_i, x_j), f_j(x_i, x_j)$ form shares of $\langle x_i, x_j \rangle$ over $\mathbb{F}$
- Reconstructing the result = sending a single element of $\mathbb{F}$
Non-Interactive Inner Product: Applications

Inner products is a simple, but very useful function:

- Biometric authentication (via Hamming distance)
- Pattern matching (via Hamming distance)
- ML (k-nearest neighbours, SVM, rule mining…)
- Linear algebra
- Similarity measure
- Simple statistics
- …
Inner products is a simple, but very useful function:

- Biometric authentication (via Hamming distance)
- Pattern matching (via Hamming distance)
- ML (k-nearest neighbours, SVM, rule mining…)
- Linear algebra
- Similarity measure
- Simple statistics
- …
Inner products is a simple, but very useful function:

- Biometric authentication (via Hamming distance)
- Pattern matching (via Hamming distance)
- ML (k-nearest neighbours, SVM, rule mining…)
- Linear algebra
- Similarity measure
- Simple statistics
- …

**Toy Example: Biometrics**

- $n$ clients and $m$ servers with a fingerprint stored.
- Ahead of time, each party publishes an encoding of its fingerprint.
Non-Interactive Inner Product: Applications

Inner products is a simple, but very useful function:
- Biometric authentication (via Hamming distance)
- Pattern matching (via Hamming distance)
- ML (k-nearest neighbours, SVM, rule mining…)
- Linear algebra
- Similarity measure
- Simple statistics
- …

Toy Example: Biometrics

- $n$ clients and $m$ servers with a fingerprint stored.
- Ahead of time, each party publishes an encoding of its fingerprint.
- Later, a client $C_i$ can authenticate to a server $S_j$ by locally computing and sending his share of the Hamming distance, a single element of $\mathbb{F}$.
Preliminaries: LPN and LWE

LPN and LWE — Primal Form

\[
\begin{pmatrix}
G \\
G
\end{pmatrix} + \text{Noise} \approx \$
\]

Random matrix  Short secret  Noise
Preliminaries: LPN and LWE

LPN and LWE — Primal Form

\[
(G, \text{Random matrix}), (G, \text{Short secret}), (\text{Noise}) \approx \mathbb{F}_2^n
\]

LPN\((\mathbb{F}_2)\):
\[G \leftarrow \mathbb{F}_2^{m \times n}, \quad \mathbb{F}_2^n, \quad \text{Ber}(\mathbb{F}_2)^n\]

LPN\((\mathbb{F}_p)\):
\[G \leftarrow \mathbb{F}_p^{m \times n}, \quad \mathbb{F}_p^n, \quad \text{Ber}(\mathbb{F}_p)^n\]
Preliminaries: LPN and LWE

LPN and LWE — Primal Form

\[
\left( \begin{array}{c} G \\ G \end{array} \right) \cdot + \approx \$
\]

Random matrix \quad Short secret \quad Noise

LPN(\mathbb{F}_2):

\[ G \leftarrow \mathbb{F}_2^{m \times n}, \quad \_ \leftarrow \mathbb{F}_2^n, \quad \_ \leftarrow \text{Ber}(\mathbb{F}_2)^n \]

‘Sparse’

LPN(\mathbb{F}_p):

\[ G \leftarrow \mathbb{F}_p^{m \times n}, \quad \_ \leftarrow \mathbb{F}_p^n, \quad \_ \leftarrow \text{Ber}(\mathbb{F}_p)^n \]

LWE(\mathbb{F}_p):

\[ G \leftarrow \mathbb{F}_p^{m \times n}, \quad \_ \leftarrow \mathbb{F}_p^n, \quad \_ \leftarrow [-B, B]^n \]

‘Small’
Preliminaries: LPN and LWE

**LPN and LWE — Dual Form**

\[ H \cdot \begin{pmatrix} \cdot \quad \cdot \quad \cdot \quad \cdot \quad + \end{pmatrix} \approx \$

LPN(\mathbb{F}_2):
\[
G \leftarrow \mathbb{F}_2^{m \times n}, \quad \mathbb{F}_2^n \leftarrow \mathbb{F}_2^n, \quad \mathbb{Ber}(\mathbb{F}_2)^n \leftarrow \mathbb{Ber}(\mathbb{F}_2)^n
\]

LPN(\mathbb{F}_p):
\[
G \leftarrow \mathbb{F}_p^{m \times n}, \quad \mathbb{F}_p^n \leftarrow \mathbb{F}_p^n, \quad \mathbb{Ber}(\mathbb{F}_p)^n \leftarrow \mathbb{Ber}(\mathbb{F}_p)^n
\]

LWE(\mathbb{F}_p):
\[
G \leftarrow \mathbb{F}_p^{m \times n}, \quad \mathbb{F}_p^n \leftarrow \mathbb{F}_p^n, \quad \mathbb{Ber}(\mathbb{F}_p)^n \leftarrow [-B, B]^n
\]

\text{‘Sparse’}
\text{‘Small’}
Preliminaries: LPN and LWE

LPN and LWE — Dual Form

\[
\begin{pmatrix}
H \\
H
\end{pmatrix}
\approx$

Random matrix

Noise

LPN($\mathbb{F}_2$): $H \leftarrow \mathbb{F}_2^{m \times n}$, $\leftarrow \mathbb{B}er(\mathbb{F}_2)^n$

LPN($\mathbb{F}_p$): $H \leftarrow \mathbb{F}_p^{m \times n}$, $\leftarrow \mathbb{B}er(\mathbb{F}_p)^n$

LWE($\mathbb{F}_p$): $H \leftarrow \mathbb{F}_p^{m \times n}$, $\leftarrow \mathbb{B}er([-B, B]^n)$

‘Sparse’

‘Small’
Alekhnovich Key Exchange

\[ H^T \cdot r_0 = \text{pk}_0 \]

\[ H \cdot s + r_1 = \text{pk}_1 \]

\[ s' \cdot \text{pk}_0 = K' \]
Alekhnovich Key Exchange

\[ H^T \cdot r_0 = p_{k_0} \]

\[ H \cdot s + r_1 = p_{k_1} \]

**Correctness**

Claim: \( \Pr[K = K'] \approx t^2/n \ll 1 \)

\[ K' = r_0^T \cdot (H \cdot s + r_1) = (H^T \cdot r_0)^T \cdot s + r_0^T \cdot r_1 \]

\[ = p_{k_0}^T \cdot s + r_0^T \cdot r_1 = K + e \]

Where \( \Pr[e = 1] \approx t^2/n \ll 1 \)
Alekhnovich Key Exchange

\[ H^\top \cdot r_0 = p_{k_0} \]

\[ H \cdot s + r_1 = p_{k_1} \]

Correctness

Claim: \( \Pr[K = K'] \approx t^2/n \ll 1 \)

\[ K' = r_0^\top \cdot (H \cdot s + r_1) = (H^\top \cdot r_0)^\top \cdot s + r_0^\top \cdot r_1 = p_{k_0}^\top \cdot s + r_0^\top \cdot r_1 = K + e \]

Where \( \Pr[e = 1] \approx t^2/n \ll 1 \)
Alekhnovich Key Exchange

\[ H^\top \cdot r_0 = p_{k_0} \]

\[ H \cdot s + r_1 = p_{k_1} \]

Correctness

Claim: Pr[\(K = K'\)] \(\approx t^2/n \ll 1\)

\[ K' = r_0^\top \cdot (H \cdot s + r_1) = (H^\top \cdot r_0)^\top \cdot s + r_0^\top \cdot r_1 \]

\[ = p_{k_0} \cdot s + r_0^\top \cdot r_1 = K + e \]

Where Pr[\(e = 1\)] \(\approx t^2/n \ll 1\)
Alekhnovich Key Exchange

\[ H^\top \cdot r_0 = pk_0 \]

\[ H \cdot s + r_1 = pk_1 \]

Correctness

Claim: \( \Pr[K = K'] \approx \frac{t^2}{n} \ll 1 \)

\[ K' = r_0^\top \cdot (H \cdot s + r_1) = (H^\top \cdot r_0)^\top \cdot s + r_0^\top \cdot r_1 = pk_0 \cdot s + r_0 \cdot r_1 = K + e \]

Where \( \Pr[e = 1] \approx \frac{t^2}{n} \ll 1 \)
Alekhnovich Key Exchange

\[ H^T \cdot r_0 = n \cdot H \cdot s + r_1 = \]

Hamming weight \( t \)

Correctness

Claim: \( \Pr[K = K'] \approx \frac{t^2}{n} \ll 1 \)

\[ K' = r_0^T \cdot (H \cdot s + r_1) = (H^T \cdot r_0)^T \cdot s + r_0^T \cdot r_1 = \]

\[ = pk_0^T \cdot s + r_0^T \cdot r_1 = K + e \]

Where \( \Pr[e = 1] \approx \frac{t^2}{n} \ll 1 \)
Alekhnovich Key Exchange

Dual LPN

Primal LPN

\[ H^T \cdot r_0 = s \cdot \cdot pk_0 \]

\[ H \cdot s + r_1 = pk_1 \]

\[ r_0 \cdot s' = pk_0 \]

\[ K' \]

\[ K \]

Security

Follows from dual LPN + primal LPN

(The proof is standard)
Embedding an Inner Product in Alekhnovich’s Key Exchange

\[ H^\top \cdot r_0 = p_{k_0} \]

\[ H \cdot s + r_1 = p_{k_1} \]

\[ r_0 \cdot K' \]

\[ s' \cdot K \]
Since the parties are computing an inner product to get $K$, could we embed an inner product computation?
Embedding an Inner Product in Alekhnovich’s Key Exchange

Since the parties are computing an inner product to get $K$, could we embed an inner product computation?

Warmup attempt: only Bob has an input

Input: $x$

Bob computes: $x - H^T \cdot r_0 = pk_0$

Alice computes: $H \cdot s + r_1 = pk_1$
Embedding an Inner Product in Alekhnovich’s Key Exchange

Since the parties are computing an inner product to get $K$, could we embed an inner product computation?

Warmup attempt: only Bob has an input

Bob computes:

Alice computes:

$s^T \cdot pk_0 = \frac{1}{2}$
Since the parties are computing an inner product to get $K$, could we embed an inner product computation?

Warmup attempt: only Bob has an input

Bob computes:

$$r_0 \cdot pk_1 = -s^t \cdot pk_0 + s^t \cdot x + e$$

Alice computes:

$$s^t \cdot pk_0$$
Embedding an Inner Product in Alekhnovich’s Key Exchange

Since the parties are computing an inner product to get $K$, could we **embed** an inner product computation?

**Warmup attempt: only Bob has an input**

Bob computes:

$$r_0 \cdot pk_0 - K' = -s^T \cdot pk_0 + s^T \cdot x + e$$

Alice computes:

$$H^T \cdot r_0 = pk_0$$

$K$ and $K'$ form (noisy) additive share of $\langle x, s \rangle$

We are making progress — but $s$ has to be random for primal LPN to hold!
Embedding an Inner Product in Alekhnovich’s Key Exchange

How to embed Alice’s input in $s$?
Embedding an Inner Product in Alekhnovich’s Key Exchange

Idea: split $H$ as $H_0 | H_1$
**Embedding an Inner Product in Alekhnovich’s Key Exchange**

Idea: split $H$ as $H_0 \ | H_1$

- **Input:** $x$
  
  - $H^\top \cdot r_0 = pk_0$
  
  - $H \cdot s = H_0 \cdot s_0 + H_1 \cdot s_1$

- **Input:** $y$
  
  - $y \cdot s_1 = s_1$

- $H_0 \cdot s_0 + H_1 \cdot s_1 = H \cdot s$

- $H^\top \cdot r_0 = pk_0$

- $H_0 \cdot s_0 + H_1 \cdot s_1 = H \cdot s$

- $y \cdot s_1 = s_1$
Embedding an Inner Product in Alekhnovich’s Key Exchange

Idea: split $H$ as $H_0 | H_1$

Input: $x$

$H^\top \cdot r_0 = pk_0$

$H \cdot s_0 + = H_0 \cdot s_0 + H_1 \cdot s_1$

Input: $y = s_1$

$K$

$s_0^\top y^+$
Embedding an Inner Product in Alekhnovich’s Key Exchange

Idea: split $H$ as $H_0 \mid H_1$

Input: $x$

$H^\top \cdot r_0 = p_{k_0}$

$H \cdot y = r_1 + p_{k_1}$

Input: $y = s_1$

$H \cdot s = H_0 \cdot s_0 + H_1 \cdot s_1$

$K' \cdot r_0 = p_{k_1}$

$K \cdot y^\top = \cdot p_{k_0}$
Embedding an Inner Product in Alekhnovich’s Key Exchange

\[ x \rightarrow H^\top \cdot r_0 = pk_0 \rightarrow H \cdot y + r_1 = pk_1 \]

Idea: split \( H \) as \( H_0 | H_1 \)

\[ K' = r_0^\top \cdot pk_s \]

\[ K = s_0^\top \cdot y^\top \cdot x + e \]

Input: \( x \)

Input: \( y = s_1 \)
Embedding an Inner Product in Alekhnovich’s Key Exchange

Input: $x$

Idea: split $H$ as $H_0 | H_1$

Security

Use primal LPN with matrix $H_0$ for Alice, and dual LPN with matrix $H^\top$ for Bob

$H^\top \cdot r_0 = pk_0$

$H \cdot s = H_0 \cdot s_0 + H_1 \cdot s_1$

$K' = -r_0 \cdot pk_x$

$K = s_0^\top \cdot y^\top + s_1^\top \cdot y^\top + e$

$y^\top \cdot x = s_0^\top \cdot y^\top \cdot x + s_1^\top \cdot y^\top \cdot x$

Input: $y = s_1$
Embedding an Inner Product in Alekhnovich’s Key Exchange

**Efficiency**

**Communication:** $2m$ from Bob and $4m$ from Alice with reasonable parameters ($m = |x| = |y|$), $(1 + \varepsilon)m$ from each party asymptotically (which is optimal).

**Computation:** cost dominated by $v \rightarrow H \cdot v$ ($\iff v \rightarrow H^T \cdot v$), can be $O(m \cdot \log m)$ (LPN with quasi-cyclic codes, standard) or even $O(m)$ (Druk-Ishai codes, slightly more exotic).

**Security**

Use primal LPN with matrix $H_0$ for Alice, and dual LPN with matrix $H^T$ for Bob.
The protocol has $t^2/n$ correctness error.

⚠️ In MPC, correctness errors translate to leakage when a ‘detectable’ error occurs: the server learns an equation $\langle v, r \rangle$ with $v$ known and $r$ the noise vector.

$\implies$ leaks $\approx N \cdot t^2/n$ linear equations in $r$ if the client interacts with $N$ servers.

$\implies$ still secure under the **LPN with leakage** assumption (equivalent to standard LPN, but with a loss)
The protocol has $t^2/n$ correctness error.

⚠️ In MPC, correctness errors translate to leakage when a ‘detectable’ error occurs: the server learns an equation $\langle v, r \rangle$ with $v$ known and $r$ the noise vector.

$\implies$ leaks $\approx N \cdot t^2/n$ linear equations in $r$ if the client interacts with $N$ servers.

$\implies$ still secure under the LPN with leakage assumption (equivalent to standard LPN, but with a loss)

### Alternatives

The above is fine when $N$ is not too large. For large $N$, or when overwhelming correctness matters (e.g. for biometric authentication), two alternatives:

1. We give an LWE-based variant with negligible error
2. We describe a way to remove errors via a sublinear-communication preprocessing phase
A Variant under LWE

\[ H^\top \cdot r_0 = s_0 + r_1 = y \]
A Variant under LWE

\[ x \in \mathbb{Z}_p \]

\[ (q/p) \cdot x \]

\[ \in \mathbb{Z}_q^{k \times n} \]

\[ H^\top \]

\[ H \cdot r_0 \]

\[ = \]

\[ y \in \mathbb{Z}_p \]

\[ \cdot s_0 \]

\[ + \]

\[ r_1 \]

\[ = \]

\[ y \cdot (q/p) \]

\[ \cdot \]

\[ p_{k_0} \]

\[ \cdot \]

\[ p_{k_1} \]
A Variant under LWE

\[ x \in \mathbb{Z}_p \]

\[ H^{\top} \cdot r_0 = p_{k_0} \]

\[ \ell \in \mathbb{Z}_q^{k \times n} \]

\[ y \in \mathbb{Z}_p \]

\[ (q/p) \cdot x \]

\[ H \cdot y \cdot (q/p) + r_1 = p_{k_1} \]

\[ K' \]

\[ r_0^\top \cdot p_{k_0} = (q/p) \cdot y^\top \cdot x \]

\[ k \]

\[ (q/p) \cdot x \]

\[ (q/p) \cdot s_0^\top \cdot y^\top \cdot 0 + e \]
A Variant under LWE

\[ x \in \mathbb{Z}_p \]

\[ H^\top \]

\[ r_0 \]

\[ y \in \mathbb{Z}_p \]

\[ e = r_0^\top \cdot r_1 \text{ with } r_0, r_1 \text{ in } [-B, B]^n \]

\[ e \leq n \cdot B^2 \]
A Variant under LWE

\[ \begin{align*}
    x \in \mathbb{Z}_p & \quad \in \mathbb{Z}_q^{k \times n} \\
    (q/p) \cdot x & \quad y \cdot (q/p) \quad r_1 & \quad = \quad y \in \mathbb{Z}_p \\
    H^\top & \quad s_0 & \quad + & \quad = \\
    r_0 & \quad \in \mathbb{Z}_p & \quad & \\
    H & \quad \cdot & \quad & \\
    \cdot & \quad s_0 & \quad + & \quad = \\
    r_0 & \quad \cdot & \quad & \\
    (q/p) \cdot y^\top & \quad 0 & \quad & \\
    (q/p) \cdot x & \quad \in \mathbb{Z}_p & \quad & \\
    (q/p) \cdot & \quad & \in \mathbb{Z}_p & \\
    s_0^\top & \quad x & \quad & \\
    y^\top & \quad 0 & \quad & \\
    x & \quad & \in \mathbb{Z}_p & \\
    r_0^\top & \quad r_1 & \quad & \\
    -K & \quad & \in \mathbb{Z}_p & \\
    K' & \quad & \in \mathbb{Z}_p & \\
    \Rightarrow e \leq n \cdot B^2 & \quad & \Rightarrow e \leq n \cdot B^2 & \\
\end{align*} \]

Rounding Lemma: if \( q/(p \cdot |e|) \geq \lambda^{o(1)} \), then \( [(p/q) \cdot K'] \mod p \) and \( [(p/q) \cdot K'] \mod p \) form additive shares of \( \langle x, y \rangle \) with proba \( 1 - \text{negl}(\lambda) \).
A Variant under LWE

\[ x \in \mathbb{Z}_p \quad \Rightarrow \quad \begin{pmatrix} (q/p) \cdot x \end{pmatrix} \begin{pmatrix} 0 \end{pmatrix} = \begin{pmatrix} H^\top \end{pmatrix} \cdot s_0 \cdot y + e \]

\[ e = r_0^\top \cdot r_1 \text{ with } r_0, r_1 \in [-B, B]^n \quad \Rightarrow e \leq n \cdot B^2 \]

Rounding Lemma: if \( q/(p \cdot |e|) \geq \lambda^{o(1)} \), then \( [(p/q) \cdot K'] \mod p \) and \( [(p/q) \cdot K'] \mod p \) form additive shares of \( \langle x, y \rangle \) with proba \( 1 - \text{negl}(\lambda) \).
A Variant under LWE

\[
\begin{align*}
x &\in \mathbb{Z}_p, \\
\frac{q}{p} \cdot x &\in \mathbb{Z}_q^{k \times n} \\
H &\cdot r_0 = s_0^{p_{k_0}} \\
H &\cdot y \cdot \frac{q}{p} \cdot r_1 = s_0^{p_{k_1}} \\
y &\in \mathbb{Z}_p
\end{align*}
\]

\[
\begin{align*}
(r_0^\top \cdot r_1 &\text{ with } r_0, r_1 \in [-B, B]^n \\
e &\le n \cdot B^2
\end{align*}
\]

Rounding Lemma: if \( q/(p \cdot |e|) \ge \lambda^{o(1)} \), then \([(p/q) \cdot K'] \mod p\) and \([(p/q) \cdot K'] \mod p\) form additive shares of \(\langle x, y \rangle\) with proba \(1 - \text{negl}(\lambda)\).

Output

\([(p/q) \cdot K'] \mod p\]

Negligible correctness error & leakage

Larger communication overhead

Output

\([(p/q) \cdot K] \mod p\)
Removing Errors via Preprocessing

Preprocessing phase

Functionality

Samples $t$-sparse vectors $(r_0, r_1)$ and set $(s_0, s_1) \leftarrow \text{Share}(\langle r_0, r_1 \rangle)$
Removing Errors via Preprocessing

Preprocessing phase

Functionality
Samples $t$-sparse vectors $(r_0, r_1)$ and set $(s_0, s_1) \leftarrow \text{Share}(\langle r_0, r_1 \rangle)$

Online phase

$0 - H^T \cdot r_0 = p_{k_0}$

$H \cdot (s_0 + y) = r_1 + p_{k_1}$
Removing Errors via Preprocessing

Preprocessing phase

Functionality

Samples $t$-sparse vectors $(r_0, r_1)$ and set $(s_0, s_1) \leftarrow \text{Share}(\langle r_0, r_1 \rangle)$

Online phase

Output

$\langle r_0, pk_1 \rangle + s_0$ = $\langle x, y \rangle - \langle r_0, r_1 \rangle + (s_0 + s_1) = \langle x, y \rangle$

Output

$\langle s, pk_0 \rangle + s_1$
Removing Errors via Preprocessing

Preprocessing phase

Functionality
Samples ℓ-sparse vectors \((r_0, r_1)\) and set \((s_0, s_1) ← \text{Share}(⟨r_0, r_1⟩)\)

Implementing the preprocessing
Removing Errors via Preprocessing

Preprocessing phase

Functionality
Samples \( t \)-sparse vectors \((r_0, r_1)\) and set \((s_0, s_1) \leftarrow \text{Share}(\langle r_0, r_1 \rangle)\)

Implementing the preprocessing

Write \( r_\sigma = \sum_{i=1}^{t} r_\sigma^{(i)} \) where the \( r_\sigma^{(i)} \) are unit vectors

\[
\langle r_0, r_1 \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} \langle r_0^{(i)}, r_1^{(j)} \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} [k_\sigma^{(i)}, k_\sigma^{(j)}] \cdot v_0^{(i)} v_1^{(j)}
\]

\( \implies s_0, s_1 \) can be securely computed using \( O(t^2 \cdot \log n) \) communication
Removing Errors via Preprocessing

Preprocessing phase

Functionality
Samples $t$-sparse vectors $(r_0, r_1)$ and set $(s_0, s_1) \leftarrow \text{Share}(\langle r_0, r_1 \rangle)$

Implementing the preprocessing
Write $r_{\sigma} = \sum_{i=1}^{t} r_{\sigma}^{(i)}$ where the $r_{\sigma}^{(i)}$ are unit vectors

$$r_{\sigma} = \begin{bmatrix} v_1 \cdots v_t \end{bmatrix}$$

Then: $\langle r_0, r_1 \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} \langle r_0^{(i)}, r_1^{(j)} \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} \lfloor k^{(i)}_{\sigma} = k^{(j)}_{\sigma} \rfloor \cdot v_0^{(i)} v_1^{(j)}$

$\Rightarrow s_0, s_1$ can be securely computed using $O(t^2 \cdot \log n)$ communication

Improved version: using LPN with regular noise brings the cost down to $O(t \cdot \log n)$
In the malicious setting, Alice and Bob must prove that $\text{pk}_0$, $\text{pk}_1$ are well-formed.
In the malicious setting, Alice and Bob must prove that $pk_0$, $pk_1$ are well-formed.
In the malicious setting, Alice and Bob must prove that $pk_0$, $pk_1$ are well-formed.

**ZK:** knows a sparse $r_0$ such that $H_0^T \cdot r_0 = pk_0^0$

**ZK:** knows sparse $r_1$ such that $\exists s, H \cdot s + r_1 = pk_1$
In the malicious setting, Alice and Bob must prove that $pk_0$, $pk_1$ are well-formed.

We introduce a new efficient ZKPoK for LPN relations, with communication $O(t \cdot \log n)$. 

**ZK:** knows a sparse $r_0$ such that $H_0^T \cdot r_0 = pk_0^0$

**ZK:** knows sparse $r_1$ such that $\exists s, H \cdot s + r_1 = pk_1$
Malicious Security

In the malicious setting, Alice and Bob must prove that pk₀, pk₁ are well-formed

\[ H^T_0 \cdot r_0 = pk_0^0 \]

\[ H^T_1 \cdot r_0 = \]

\[ H \cdot s_0 + y = pk_1 \]

We introduce a new efficient ZKPoK for LPN relations, with communication \( O(t \cdot \log n) \)

**Idea**

Recent works on *pseudorandom correlation generator* show how to distribute shares of \( \Delta \cdot r \) where \( r \in \mathbb{F}_p^n \) is \( t \)-sparse, and \( \Delta \in \mathbb{F}_{p^k} \), via *function secret sharing* (which exist under OWF)

\( \implies \) we use these techniques to authenticate the noise vectors with a MAC \( \Delta \)

\( \implies \) we use the MAC to check the correct computation of the pk’s, *al la* SPDZ.
Thank you for your attention!

Questions?