MPC with Silent Preprocessing via Pseudorandom Correlation Generators

Lisa Kohl

Based on joint works with Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Peter Rindal, and Peter Scholl
Secure multi-party computation (MPC)
[Yao86; GMW87; BGW88; CCD88]

Goal: Parties learn $f(a, b)$ and nothing more
Secure MPC with preprocessing

Secure MPC with preprocessing [Beaver91]

**Correlated randomness**

- Fast online phase, security against dishonest majority
- Preprocessing expensive (communication & storage)
Pseudorandom correlation generator (PCG) [BCGI18; BCGIKS19]

- Fast online phase
- Preprocessing expensive (communication & storage)

Correctness: $R_0 \sim R_1$

Security: $(k_0, R_1) \approx c(k_0, [R_1 | R_0 \sim R_1])$

$R_0 \leftarrow \text{Expand}(k_0)$

$R_1 \leftarrow \text{Expand}(k_1)$

Short correlated seeds

Correlated randomness
Pseudorandom correlation generator (PCG)
[BCGI18; BCGIKS19]

\[ R_0 \leftarrow \text{Expand}(k_0) \]
\[ R_1 \leftarrow \text{Expand}(k_1) \]

Correctness: \( R_0 \sim R_1 \)
Pseudorandom correlation generator (PCG)

[BCGI18; BCGIKS19]

\[ k_0 \]
\[ R_0 \leftarrow \text{Expand}(k_0) \]
\[ R_0 \]

\[ k_1 \]
\[ R_1 \leftarrow \text{Expand}(k_1) \]
\[ R_1 \]

Security: \((k_0, R_1) \approx_c (k_0, [R_1 | R_0 \sim R_1])\)
Secure MPC with *silent* preprocessing

[BCGIKS19]

- **Short correlated seeds**
- **Correlated randomness**

\[
R_0 \leftarrow \text{Expand}(k_0) \\
R_1 \leftarrow \text{Expand}(k_1)
\]

- **Setup with sublinear communication & storage**
- **malicious security at little extra cost**

---

**SILENT**
Generic construction of PCGs

General additive correlations:

\[ R_0 + R_1 = f(X) \]

Feasibility: PRG + Homomorphic secret sharing

\[ k_0 \leftarrow \text{Eval}_{f \circ \text{PRG}}(k_0) \]

\[ k_1 \leftarrow \text{Eval}_{f \circ \text{PRG}}(k_1) \]
## Landscape of PCGs

<table>
<thead>
<tr>
<th>Region</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Gentryland”</strong></td>
<td>LWE+: General additive [BCGIKS19]</td>
<td></td>
</tr>
<tr>
<td><strong>“Cryptomania”</strong></td>
<td>DDH + PRG*: Log-space [BCGI017]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LWE + PRG*: Bounded depth* [BCGIKS19]</td>
<td></td>
</tr>
<tr>
<td><strong>“Lapland”</strong></td>
<td>LPN: Vector OLE [BCGI18]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ring-LPN: OT, Constant-degree [BCGIKS19]</td>
<td></td>
</tr>
<tr>
<td><strong>“Minicrypt”</strong></td>
<td>OWF: Linear [GI99; CDI05]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Truth tables [BCGIKS19]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*low-degree *concretely efficient</td>
<td></td>
</tr>
</tbody>
</table>
Learning with errors vs. learning parity with noise

**LWE:**
\[ p > 2 \]
\[ s \text{ over } \mathbb{Z}_p \]
\[ \| e \|_\infty \text{ small} \]

**LPN:**
\[ p = 2 \text{ (here: } p \geq 2) \]
\[ s \text{ over } \mathbb{Z}_p \]
\[ \text{HW}(e) \text{ small} \]
Cryptography from LWE vs. LPN
Cryptography from LWE vs. LPN

- LPN
- LWE
- OWF
- PRG
- CRH
- PKE
- OT
- NIKE
- Additive HE
- PCG for OT
- PCG for OLE
- low noise
- LWE
- LPN
A simple PRG from LPN

**LPN:**
- generator matrix $G$
- limited to quadratic stretch

**Dual-LPN:**
- parity check matrix $H$
- arbitrary polynomial stretch
Why LPN is a perfect match for PCGs

- Sparse vector can be distributed via compressed secret shares
- LPN assumption is linear $\approx$ homomorphic properties
How to distribute a sparse vector efficiently

Point Function: $F^{\alpha}: \{1, \ldots, N\} \rightarrow \mathbb{F}_{2^\lambda}$, $F^{\alpha}(x) = \begin{cases} y, & \text{if } x = \alpha \\ 0, & \text{else} \end{cases}$

Distributed Point Function:

Efficient constructions from OWFs [GI14; BGI16]
Efficient distributed setup [Ds17]
Part I: PCG for oblivious transfer from LPN
Oblivious transfer (OT)  
[Rab81; EGL85]

Security: Alice learns only $r_b$, Bob doesn’t learn $b$

GMW Protocol: Secure MPC with 2 OTs per AND-Gate

Problem: OT is expensive ("public-key primitive")
OT extension

OT extension: Few base OTs + “cheap crypto” [Bea96; IKNP03]

Silent OT extension: Local expansion [BCGIKS19; BCGIKRS19]
Comparison of OT extension protocols

128-bit security

<table>
<thead>
<tr>
<th>Reference</th>
<th>Rounds</th>
<th>Comm. per random OT</th>
<th>Silent</th>
<th>Active</th>
<th>Based on</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Bea96]</td>
<td>2</td>
<td>poly</td>
<td>✗</td>
<td>✗</td>
<td>OWF</td>
</tr>
<tr>
<td>[IKNP03; ALSZ13; KOS15]</td>
<td>3*</td>
<td>128</td>
<td>✗</td>
<td>✗</td>
<td>crh</td>
</tr>
<tr>
<td>[KK13] (short strings)</td>
<td>3</td>
<td>≈ 78</td>
<td>✗</td>
<td>✗</td>
<td>crh</td>
</tr>
<tr>
<td>[BCGIKS19]</td>
<td>log N</td>
<td>0 – 3</td>
<td>✓</td>
<td>✗</td>
<td>LPN, crh**</td>
</tr>
<tr>
<td>[BCGIKRS19]</td>
<td>2*</td>
<td>0.1</td>
<td>✓</td>
<td>✓</td>
<td>LPN, crh**</td>
</tr>
</tbody>
</table>

*Fiat-Shamir for active security, **correlated-input secure hash function

[GMMM18]: RO ⇔ 2-round OT extension
Comparison of OT extension protocols

128-bit security

<table>
<thead>
<tr>
<th>Reference</th>
<th>Rounds</th>
<th>Comm. per random OT</th>
<th>Silent</th>
<th>Active</th>
<th>Based on</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Bea96]</td>
<td>2</td>
<td>poly</td>
<td>x</td>
<td>x</td>
<td>OWF</td>
</tr>
<tr>
<td>[IKNP03; ALSZ13; KOS15]</td>
<td>3*</td>
<td>128</td>
<td>x</td>
<td>✓</td>
<td>crh</td>
</tr>
<tr>
<td>[KK13] (short strings)</td>
<td>3</td>
<td>≈ 78</td>
<td>x</td>
<td>x</td>
<td>crh</td>
</tr>
<tr>
<td>[BCGIKS19]</td>
<td>log N</td>
<td>0 – 3</td>
<td>✓</td>
<td>x</td>
<td>LPN, crh**</td>
</tr>
<tr>
<td>[BCGIKRS19]</td>
<td>2*</td>
<td>0.1</td>
<td>✓</td>
<td>✓</td>
<td>LPN, crh**</td>
</tr>
</tbody>
</table>

*Fiat-Shamir for active security, **correlated-input secure hash function

- Semi-honest 2-PC w/ 4.2 bits per AND, 30× less than [DKSSZZ17]
- Improves PSI, malicious MPC
- Useful for non-interactive secure comp. [IKOPS11; AMPR14; MR17]
Correlated OT

$\in \mathbb{F}_2$

$\in \mathbb{F}_{2^\lambda}$

$r_i, r_i + \Delta$

$r_i + \Delta \cdot b_i$

Correlated OT + correlation robust hash function $\Rightarrow$ OT [IKNP03]

As vectors: $\triangleq$ Subfield vector oblivious linear evaluation

$\Delta \cdot b = r + \Delta \cdot b + r$
Overview: PCG for correlated OT

[BCGIKS19]

Idea:

1. Via distributed point functions:

\[ k_0 + k_1 = \Delta \]

2. Via addition:

\[ \Delta + \Delta + \Delta = \Delta \cdot b \]

3. Via LPN:
1a. Towards 2-round setup

Problem: DPF require $\log N$ rounds for distributed setup!

Observation:
- Receiver knows $b$
- Receiver knows the point $\alpha$, where PF $\neq 0$
- Puncturable pseudorandom functions sufficient!
1b. Puncturable pseudorandom function

[BGI13; BW13; KPTZ13]

Puncturable PRF (PPRF):

\[ F_k : \{1, \ldots, N\} \rightarrow \mathbb{F}_{2^\lambda} \]

- \( k \) \( \rightsquigarrow \) \( F_k(x) \) for all \( x \)
- \( k^* \) \( \rightsquigarrow \) \( F_k(x) \) for all \( x \neq \alpha \)

Via GGM:

\[ k \]

\[ 1 \quad \alpha \quad \cdots \quad N \]

\(|k| = \lambda, \ |k^*| = \lambda \log N\]
1c. PCG for unit vector via PPRF

How to set up $k^*$, $k$?

$\alpha$, $k^*$, $F_k(\alpha) + \Delta$

$k$, $\Delta$
1d. 2-Round setup for unit vector

[SGRR19; BCGIKRS19]

**Strategy:** (based on [Ds17])

- Sender chooses $k$
- Receiver receives $k^*$ via chosen OTs:

**Note:** OTs can be executed *in parallel!*
2. From unit to sparse vectors
[BCGI18; BCGIKS19]

Repeat $t$ times:

Alternative: Concatenation $+$ LPN with *regular* noise
3. From sparse to pseudorandom vectors

[BCGI18; BCGIKS19]

Main challenge: Parity check matrix is big!
- use quasi-cyclic codes \(\sim\) multiplication in \(\tilde{O}(N)\)

Security
- Similar to PQ cryptosystems BIKE, HQC [AAB+19; ABB+19]
PCG for correlated OT from LPN - Recap

\[ \alpha_i, \quad k_i^*, \quad F_{k_i}(\alpha_i) + \Delta \]

\[ r \quad + \quad \Delta \cdot b \]

\[ k_i, \Delta \]

\[ r \]
From correlated OT to chosen OT

1. Break correlations:
   ▶ *Locally* apply crh [IKNP03]

   \[ \Rightarrow \text{MPC with 2-round silent preprocessing} \]

2. Derandomization:
   ▶ Depends only on \( b \)
   ▶ Can be sent along with first message

   \[ \Rightarrow \text{2-round OT extension} \]
Runtimes (ms) for 10 million random OTs

[BCGIKRS19]

[IKNP03] vs 2-round silent vs 3-round hybrid

- Total communication: 160 MB vs 145 kB vs 127 kB
Part II: PCGs for OLE from LPN and ring-LPN
Oblivious linear evaluation (OLE)

- Generalization of OT to $\mathbb{F}_p$
- 2 OLEs can be locally transformed into a multiplication triple

\[
d = ab + c
\]
Towards PCG for OLE from LPN

Idea: Rewrite $a \ast b$ and use linearity of LPN

Via LPN:
PCG for OLE via LPN

[BCGIKS19, BCGIKS20]

Via DPF:

Problem: Dimension (\(\sim\) computational cost) quadratic in \(N\)
A different perspective
[BCGIKS20]

Observations:

- Generalizes to more dimensions
- Better efficiency via choosing $H$ such that $H \ast H$ compressible
More efficient PCG for OLE from ring-LPN

[a(X) \cdot s(X) + e(X) \approx_c u(X)] over \mathbb{Z}_p[X]/\varphi(X)

If \varphi(X) (of degree N) fully splits over \mathbb{Z}_p[X]:

AV \quad V \quad \approx_c \quad $
More efficient PCG for OLE from ring-LPN

If $\psi(X)$ (of degree $N$) fully splits over $\mathbb{Z}_p[X]$:

$\sim N$ OLEs over $\mathbb{Z}_p$ in $\tilde{O}(N)$ computation time
Efficiency of our PCG construction for OLE

[BCGIKS20]

To generate 1 Mio OLEs over $\mathbb{Z}_q$ ($q$ composite of 62-bit primes):

<table>
<thead>
<tr>
<th>Reference</th>
<th>Amount</th>
<th>Seed size</th>
<th>Communication</th>
<th>OLEs/second</th>
</tr>
</thead>
<tbody>
<tr>
<td>[KPR18]</td>
<td>32 MB</td>
<td>32 MB</td>
<td>&gt; 1 GB</td>
<td>30 K</td>
</tr>
<tr>
<td>[BCGIKS19]</td>
<td>17 GB</td>
<td>3 GB</td>
<td>6 GB</td>
<td>6 K*</td>
</tr>
<tr>
<td>[BCGIKS20]</td>
<td>32 MB</td>
<td>1.25 MB</td>
<td>7 MB</td>
<td>100 K*</td>
</tr>
</tbody>
</table>

*expansion only, estimated costs

- Setup with malicious security
- Generalizes to authenticated multiplication triples at $\approx \times 2$ cost!
Conclusion

PCGs for OT from LPN [BCGIKS19; BCGIKRS19]
- Random OT: *practical*, almost zero communication
- 2-Round OT extension (malicious security, implementation)

PCGs for OLE [BCGIKS20]
- More efficient instantiation based on *fully splittable ring-LPN*

Open problems/ Ongoing work:
- Optimize OT: Better codes
- Efficient PCGs for more correlations
- Better understanding of LPN-flavored assumptions

Thank you!