CS 65500
Advanced Cryptography

Lecture 20: Pseudorandom Correlation Generator

Instructor: Aarushi Goel

Spring 2025

Agenda

- → MPC with Pre-processing
- → Pseudorandom correlations
- → Pseudorandom Correlation Generator (PCG)
- → LPN assumption
- → MPFSS
- → PCG for VOLE correlation

Secure Computation with Preprocessing

→ Earlier in this course we saw that Beaver triples can be used to reduce communication in existing secret sharing based secure multiparty computation protocols

How does one obtain these Beaver triples?

→ As we saw in HW4, these Beaver triples could be generated by the parties using any secure multiparty computation protocol.

→ However, that would require additional communication.

→ Reducing communication in MPC using Beaver triples, but requiring a lot of communication to generate Beaver triples is not ideal.

Today: How to efficiently (i.e., with low communication) generate Beaver triples.

Beaver Triples

→ Recall that Beaver triples are tuples of the form (a, b, c) where a, b are uniformly sampled, $c = a \cdot b$ and it each party receives a secret share of a, b & c.

→ Observe that, it also suffices for a, b to be <u>pseudorandom</u> values as opposed to uniformly sampled ones.

→ We will only consider the case where parties obtain <u>additive</u> secret shares

→ In fact, for simplicity, we only focus on the <u>two-party</u> setting.

Vector Oblivious Linear Evaluation (VOLE):

→ Vector oblivious linear evaluation (VOLE) correlation is a two-party correlation of the form $\vec{u} \cdot x + \vec{v} = \vec{w}$, where $\vec{u}, \vec{v}, x$ are random values. One party gets $(\vec{u}, \vec{v})$, the other party gets $(\vec{w}, x)$.

→ We will first focus on designing an efficient mechanism for generating pseudorandom VOLE correlations (i.e., where u, v, x are pseudorandom), and then ~~discuss~~ how those ideas can be extended to generate Beaver triples. → question in the next HW! :)

Few Correlations → Many Correlations

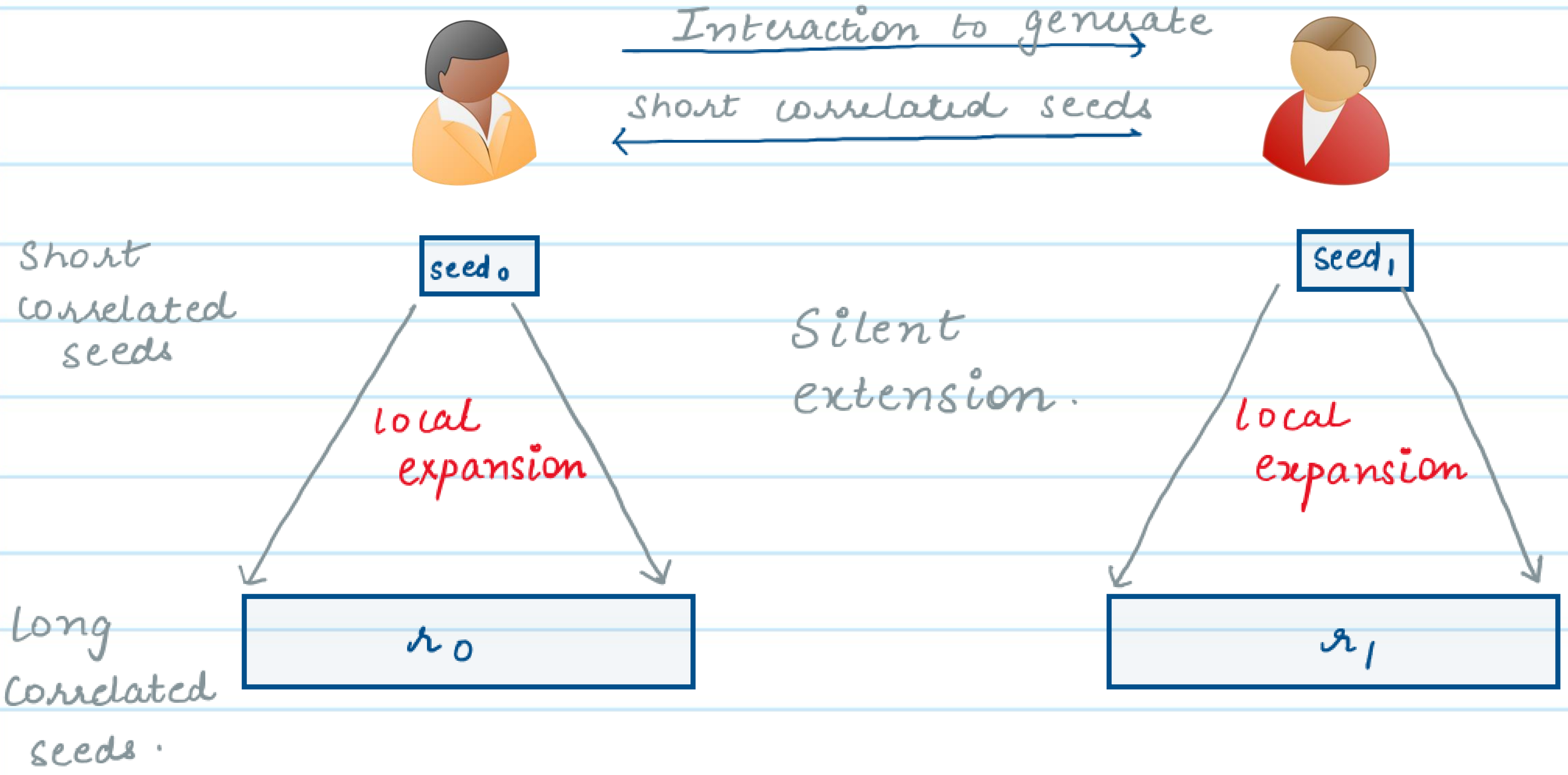
→ Pseudorandom generators (PRGs) allow local (and deterministic) expansion of a short random seed into a long pseudo-random string.

$$\boxed{\text{seed}} \xrightarrow[\text{expansion}]{\text{local}} \boxed{\text{long, pseudorandom string}}$$

→ We will use a similar idea to expand a set of few correlated random seeds into many correlated pseudorandom strings.

→ Importantly, given the set of few correlated random seeds, we want parties to be able to locally expand them into many correlated pseudorandom strings.

Pseudorandom Correlation Generator (PCG)



Pseudorandom Correlation Generator (PCG)

→ Intuitively speaking, we want that $seed_b$ observed by Party b together with the expanded second output $Expand(seed_{1-b})$ are indistinguishable from $seed_b$ together with random output of Party $1-b$ conditioned on $Expand(seed_b)$ in a perfect VOLE correlation.

→ For security against party 0, this essentially means $seed_0$ should hide x.

→ For security against party 1, this means that given $(\vec{u}, \vec{v}, seed_1)$, an adversary cannot tell whether $\vec{u}, \vec{v}$ were generated by expanding $seed_0$ or sampled randomly such that $\vec{u} \cdot x + \vec{v} = Expand(seed_0)$

Defining PCG for VOLE Correlation

Definition: A pseudorandom VOLE generator is a pair of PPT algorithms (Setup, Expand) defined as follows:

→ Setup$(1^\lambda, \mathbb{F}, n, x)$: Outputs $(seed_0, seed_1)$, where $seed_1$ includes x.

→ Expand $(\sigma, seed_\sigma)$: if $\sigma = 0$, outputs $(\vec{u}, \vec{v}) \in \mathbb{F}^n \times \mathbb{F}^n$. If $\sigma = 1$, output $\vec{w} \in \mathbb{F}^n$.

These algorithms should satisfy the following:

* **Correctness:** for any $\mathbb{F}$ & any $x \in \mathbb{F}$, let $(seed_0, seed_1) \leftarrow$ Setup$(1^\lambda, \mathbb{F}, n, x)$, $(\vec{u}, \vec{v}) \leftarrow$ Expand$(0, seed_0)$, $\vec{w} \leftarrow$ Expand$(1, seed_1)$. Then $\Pr[\vec{u} \cdot x + \vec{v} = \vec{w}] = 1$

* **Security:** For any n.u. PPT adversary A, it holds that:

1. $\forall x, x' \in \mathbb{F}$, $\{seed_0 \mid (seed_0, seed_1) \leftarrow \text{Setup}(1^\lambda, \mathbb{F}, n, x)\}$

$$\approx_c \{seed_0 \mid (seed_0, seed_1) \leftarrow \text{Setup}(1^\lambda, \mathbb{F}, n, x')\}$$

2. $\forall x \in \mathbb{F}$, let $(seed_0, seed_1) \leftarrow$ Setup$(1^\lambda, \mathbb{F}, n, x)$.

$$\{(\vec{u}, \vec{v}, seed_1) \mid (\vec{u}, \vec{v}) \leftarrow \text{Expand}(0, seed_0)\} \approx_c \left\{(\vec{u}, \vec{v}, seed_1) \;\middle|\; \begin{array}{l} \vec{u} \xleftarrow{\$} \mathbb{F}^n,\ \vec{v} = \vec{w} - \vec{u} \cdot x \\ \vec{w} \leftarrow \text{Expand}(1, seed_1) \end{array}\right\}$$

Learning Parity with Noise (LPN) Assumption

The LPN assumption states that the following two distributions of (A, b) are computationally indistinguishable:

1) Let $A \in \mathbb{F}^{K\times q}$ be uniformly sampled and $b \in \mathbb{F}^{1\times q}$ be defined as

$$\underbrace{\boxed{}}_{b\,\in\,\mathbb{F}^q} = \underbrace{\boxed{}}_{s\,\in\,\mathbb{F}^K} \; \underbrace{\boxed{}}_{A\,\in\,\mathbb{F}^{K\times q}} + \underbrace{\boxed{}}_{e\,\in\,\mathbb{F}^q} \qquad K \leq q.$$

s: uniformly sampled

e: Random sparse vector

i. each $x \in e$ is 0 with prob $(1-r)$ & a random non zero value with prob (r)

2) Let $A \in \mathbb{F}^{K\times q}$ & $b \in \mathbb{F}^q$ be uniformly sampled

Syndrome Decoding Problem

This is a dual version of LPN which states that following two distributions of (B, b) are computationally indistinguishable:

1) Let $A \in \mathbb{F}^{k \times q}$ be uniformly sampled, $B \in \mathbb{F}^{q \times (q-k)}$ be a full-rank matrix such that

$$\boxed{A \in \mathbb{F}^{k \times q}} \quad \boxed{B \in \mathbb{F}^{q \times (q-k)}} = \underset{(q-k)}{\boxed{0 \text{ matrix}}} \; k$$

B is also called the parity check matrix

Let $e \in \mathbb{F}^{q}$ be a sparse random vector as defined on the previous slide. Let $b = (e \cdot B) \in \mathbb{F}^{q-k}$

2) Let $B \in \mathbb{F}^{q \times (q-k)}$ be as defined above and $b \in \mathbb{F}^{q \times k}$ be sampled uniformly.

Syndrome Decoding $\equiv$ LPN assumption.

→ Syndrome decoding problem is equivalent to the regular LPN assumption.

for any $s \in \mathbb{F}^k$:

$$
\begin{aligned}
e \cdot B &= (s \cdot A + e) B \\
&= s \cdot A \cdot B + e \cdot B \\
&= s \cdot 0 + e \cdot B
\end{aligned}
$$

→ Syndrome decoding problem is also called the dual-LPN assumption.

Attacks on the LPN Problem

→ LPN assumption is used extensively in the design of public-Key crypto systems.

→ However, there are a few cryptanalytic attacks known for this assumption.

→ Some of these attacks can infact be used to recover the $\vec{s}$, while others only help distinguish $b = s \cdot A + e$ from a random vector.

→ Therefore, parameters K, q, r must be chosen to ensure that these attacks are not feasible in polynomial time.

Multipoint Function Secret Sharing (MPFSS)

→ In the last class we saw a construction of FSS for point functions of the form $f_{\alpha,\beta}(x) = \begin{cases} \beta, & \text{if } x=\alpha \\ 0, & \text{otherwise} \end{cases}$

→ A multipoint function is one that outputs non-zero values on a set of pre-determined inputs & 0 on all other inputs. e.g.,

$$f_{\alpha_1 \ldots \alpha_k, \beta_1 \ldots \beta_k}(x) = \begin{cases} \beta_i, & \text{if } x=\alpha_i \\ 0, & \text{otherwise} \end{cases}$$

→ An FSS for point functions can be trivially extended to obtain an FSS for multipoint functions — with a k time blow-up in the share size.

There are also other more efficient ways to design MPFSS

Constructing PCGs for VOLE correlations

$(\vec{u} \cdot x + \vec{v} = \vec{w})$

P_0 $\quad$ P_1

Let $A \in \mathbb{F}^{k \times q}$ be a uniformly sampled matrix.

Let $B \in \mathbb{F}^{q \times (q-k)}$ be the parity check matrix corresponding to A.

Let (MPFSS.Gen, MPFSS.Eval, MPFSS.FullEval) be an MPFSS scheme

for computing output shares on all inputs.

→ Setup$(1^\lambda, \mathbb{F}, q, k, x)$: Sample a random size-$t$ subset S of $[q]$, and a random vector $\vec{y} \xleftarrow{\$} \mathbb{F}^t$. Let $s_1 < \cdots < s_t$ denote the elements in S. Compute $(K_0, K_1) \leftarrow$ MPFSS.Gen$(1^\lambda, f_S, x\vec{y})$. Output $\text{seed}_0 = (\mathbb{F}, q, k, K_0, S, \vec{y})$, $\text{seed}_1 = (\mathbb{F}, q, k, K_1, x)$

→ Expand$(0, seed_0)$: Parse $seed_0 = (\mathbb{F}, K, q, K_0, S, \vec{y})$.
Set $\vec{\mu} \leftarrow spred_n(S, \vec{y})$. Compute $\vec{v}_0 \leftarrow$ MPFSS.FullEval$(0, K_0)$.
Compute $\vec{u} = \vec{\mu} \cdot B$, $\vec{v} = -\vec{v}_0 \cdot B$.
Output $(\vec{u}, \vec{v})$

→ Expand$(1, seed_1)$: Parse $seed_1 = (\mathbb{F}, K, q, K_1, x)$.
Compute $\vec{v}_1 \leftarrow$ MPFSS.FullEval$(1, K_1)$. Compute $\vec{w} = \vec{v}_1 \cdot B$.
Output $(x, \vec{w})$.

Correctness & Security

→ Correctness:

$$\vec{u}\cdot x+\vec{v} = (\vec{\mu}\cdot x-\vec{v}_0)\cdot B$$
$$= (\vec{\mu}\cdot x+\vec{v}_1-\vec{\mu}\cdot x)\cdot B$$
$$= \vec{v}_1\cdot B = \vec{w}$$

→ Security:

1 Security against party 0: Recall that here we want to show that for any $x, x' \in \mathbb{F}$, an adversary given seed_0 cannot tell whether it was generated using x or x'.

seed_0 in this construction comprises of K_0, $\underbrace{\vec{y}, S}_{\text{independent of } x}$.

The only part that depends on x is the MPFSS Key K_0. Privacy of x follows from secrecy of the FSS scheme.

2. <u>Security against Party 1</u>: Recall that here we want to show that for any $x \in \mathbb{F}$: given $(\vec{u}, \vec{v}, \text{seed}_1)$, the adversary cannot tell whether $\vec{u}, \vec{v}$ were obtained by expanding seed_0 or by sampling $\vec{u}$ at random & then computing $\vec{v} = \vec{w} - \vec{u} \cdot x$.

Seed_1 in this construction includes K_1, x. Intuitively speaking, K_1 does not reveal any information about $\vec{y}, S$. All that remains is to show $(\vec{u}, \vec{v}) \leftarrow \text{Expand}(0, \text{seed}_0)$ is indistinguishable from $(\vec{u}' \xleftarrow{\$} \mathbb{F}^{(q-K)}, \vec{v}' \leftarrow \text{Expand}(1, \text{seed}_1) - \vec{u}' \cdot x)$.

$\vec{v}' = \vec{v}_1 \cdot B - \vec{u}' \cdot x$. Hence, it suffices to show $\vec{v}_1 \cdot B$ is indistinguishable from a random string $\in \mathbb{F}^{(q-K)}$

Observe that $\vec{v}_1$ is a uniformly sampled t-sparse vector. Hence from the dual LPN assumption it follows then $\vec{v}_1 \cdot B$ is indistinguishable from a random string $\in \mathbb{F}^{(q-K)}$

LPN Parameters

The dual LPN assumption that we rely on is equivalent to the regular LPN assumption where $A \in \mathbb{F}^{k \times q}$, $s \in \mathbb{F}^{k}$, $e \in \mathbb{F}^{q}$, $b \in \mathbb{F}^{q}$, $r = t/q$. The best attacks in this setting are:

* Low-weight parity check attack: It takes time $O\left(\left(\frac{q}{q-k}\right)^{t}\right)$. This is exponentially large for $t = O(\lambda)$

* Gaussian-elimination attack: This takes time $O\left(\left(1-\frac{t}{q}\right)^{k}\right)$.

$$= \left(1-\frac{t}{q}\right)^{k} = \left(1-\frac{t/q \cdot (k)}{(k)}\right)^{k}$$

$$\approx e^{kt/q} \approx e^{O(t)}$$

When $q = \Theta(k)$

$e^{O(t)}$ is exponentially large for $t = O(\lambda)$

* Information Set Decoding attack: This takes time $O\left(\left(1-\frac{q-K}{q}\right)^t\right)$

$$\left(1-\frac{q-K}{q}\right)^t = \left(\left(1-\frac{q-K}{q}\right)^q\right)^{t/q}$$

$$\approx \left(e^{q-K}\right)^{t/q} \approx e^{O(t)} \quad \text{when} \quad q = \Theta(K)$$

$e^{O(t)}$ is exponentially large for $t = O(\lambda)$

$\Rightarrow$ As long as $t = O(\lambda)$ & $q = \Theta(K)$, the exact values of q & K can be chosen arbitrarily.

Expansion

This PCG construction allows expanding t-correlated seeds into q pseudorandom correlated VOLEs.
We can choose $t = O(\lambda)$ and $q = poly(\lambda)$ to get arbitrary polynomial expansion.

* The setup algorithm can be run by the two parties using any generic secure computation protocol. Communication needed for this protocol will be sublinear in q. After getting $seed_0$ and $seed_1$, they can locally get q VOLE correlations without further interaction.