More advanced preprocessing

Peeter Laud

November 2, 2022
Multiplication triples

- \((\lbrack a \rbrack, \lbrack b \rbrack, \lbrack c \rbrack)\), where \(c = ab\)

- In additive secret sharing, \(\lbrack a \rbrack = \langle a \rangle = (a_1, \ldots, a_n)\)

- In SPDZ, \(\lbrack a \rbrack = (\langle a \rangle, \langle MAC(a) \rangle)\)

- Computing \(\lbrack c \rbrack\) can be done by two-party multiplications
  - Turn \(a_i\) and \(b_j\) into shares of \(a_i b_j\)

- Recall that preprocessing finishes with the correctness check
  (all computations over some ring/field)
Oblivious Linear Evaluation (OLE)

- “Sender” inputs $u, v \in \mathbb{F}$
- “Receiver” inputs $x \in \mathbb{F}$
- “Receiver” gets $w \in \mathbb{F}$, such that $v + w = ux$
Random Oblivious Linear Evaluation (R-OLE)

- “Sender” gets $u, v \in \mathbb{F}$
- “Receiver” gets $x, w \in \mathbb{F}$, such that $v + w = ux$

**Exercise.** Construct OLE from R-OLE. No crypto (i.e. OWFs and like) is needed.
Vector Oblivious Linear Evaluation (VOLE)

- “Sender” inputs $\vec{u}, \vec{v} \in \mathbb{F}^n$
- “Receiver” inputs $x \in \mathbb{F}$
- “Receiver” gets $\vec{w} \in \mathbb{F}^n$, such that $\vec{v} + \vec{w} = \vec{u} \cdot x$
  - Addition and scalar multiplication: pointwise
Oblivious transfer

Def: OT, R-OT
From OT to OLE

- Receiver splits $x$ into bits: $x = \sum_{i=0}^{m-1} 2^i \cdot x_i$
- Sender does a “weighted additive sharing” of $v$:
  - Pick $v_0, \ldots, v_{m-1} \leftarrow F$, such that $v = \sum_{i=0}^{m-1} 2^i \cdot v_i$
  - For $i \in \{0, \ldots, m-1\}$, run OT:
    - Sender (as Sender) inputs $-v_i, u - v_i$
    - Receiver inputs $x_i$
  - Receiver obtains $w_i = ux_i - v_i \ (i \in \{0, \ldots, m-1\})$
  - Receiver defines $w \leftarrow \sum_{i=0}^{m-1} 2^i \cdot w_i$

Note that for VOLE, still need only $m$ OT-s (with longer messages)
Active security?

- Check that $w_i = x_i u - v_i$ for all $i$. Open a random linear combination. Leaks a bit
  - Combine one more equation. Use random $x_m, v_m$; run OT
Selective failure

- Sender can test whether $x_i = 0$. If $x_i = 1$, then aborts.
- Selective failure is less of a problem under definitions of covert security.
- For MPC preprocessing, we only care about active privacy anyway.
Function secret sharing (FSS)

Consider a “private” function $f : X \rightarrow \mathbb{F}$. ($\mathbb{F}$ is typically $\mathbb{Z}_p$; but in our construction, $\{0, 1\}^k$)

**Signature of FSS**

- **Gen**("$f$") $\mapsto (K_0, K_1)$, where a single $K_i$ “reveals nothing” about $f$
- Let $x \in X$ and $i \in \{0, 1\}$. Eval($i, K_i, x$) $\mapsto y_i \in \mathbb{F}$, s.t. $y_0 \oplus y_1 = f(x)$

- **Gen** is run by a trusted dealer. Eval($i, \ldots$) is run by $i$-th party
- In real protocols, Gen is run under 2PC / MPC
Distributed point functions (DPF)

Let $a \in X$, $b \in \mathbb{F}$

$$f_{a,b}(x) := \begin{cases} b, & \text{if } x = a \\ 0, & \text{otherwise} \end{cases}$$

(a point function)
Consider $f$ as a vector: $f \in \mathbb{F}^X$. Additively share it among two parties:

- $K_i$: the entire $i$-th share
- evaluation by $P_i$ on $x$: look up $x$-th entry in $K_i$

**Notice:** for a DPF, $K_0$ and $K_1 \in \mathbb{F}^X$ are equal to each other in all but a single entry
Level-1 construction for DPF

- Let $X \equiv X' \times X''$. Let $G$ be a PRG, with seeds from $F$
  - Let $G$ expand from $F$ to $F^{X''}$ (with random access)
- A DPF $f_{a,b}$ is now $f_{(a',a''),b}$

$\text{Gen}(f_{(a',a'')},b)$ is the following

- Let $\vec{L}_0, \vec{L}_1 \leftarrow F^{X'}$
  - Let them be equal everywhere, except for the $a'$-th entry
- Let $\vec{C}_0, \vec{C}_1 \leftarrow F^{X''}$: $\vec{C}_0 \oplus \vec{C}_1 = G(L_0[a']) \oplus G(L_1[a']) \oplus b \cdot \vec{e}_{a''}$
- Let $\vec{B}_0, \vec{B}_1 \leftarrow \{0,1\}^{X'}$
  - Let them be equal everywhere, except for the $a'$-th entry
- Let $K_0 = (\vec{L}_0, \vec{B}_0, \vec{C}_0, \vec{C}_1)$ and $K_1 = (\vec{L}_1, \vec{B}_1, \vec{C}_0, \vec{C}_1)$

Eval($i, (\vec{L}_i, \vec{B}_i, \vec{C}_0, \vec{C}_1), (x', x'')$) returns $(G(L_i[x']) \oplus C_{B_i[x']})[x'']$
Level-$n$ construction for DPF

The pair $(\text{zip}(\vec{L}_0, \vec{B}_0), \text{zip}(\vec{L}_1, \vec{B}_1)) \in ((\mathbb{F} \times \{0, 1\})^{X'})^2$ looks like an output of level-0 construction

- Let $s \xleftarrow{\$} \mathbb{F}$. Let $(\sigma_0, \sigma_1) \leftarrow \text{Gen}_{n-1}(f_{a'}, s\|1)$
- Let $s_0\|t_0 \leftarrow \text{Eval}_{n-1}(0, \sigma_0, a')$ and $s_1\|t_1 \leftarrow \text{Eval}_{n-1}(1, \sigma_1, a')$

$b \cdot \vec{e}_{a''}$ is a point function: $f_{a'', b} : X'' \rightarrow \mathbb{F}$

- Let $(\rho_0, \rho_1) \leftarrow \text{Gen}_{n-1}(f_{a''}, b)$. Let $\alpha = |\rho_0| = |\rho_1|
- Let $G'$ expand from $\mathbb{F}$ to $\{0, 1\}^\alpha$
- Let $\tilde{C}_{t_0} \leftarrow G'(s_0) \oplus \rho_0$ and $\tilde{C}_{t_1} \leftarrow G'(s_1) \oplus \rho_1$
- $\text{Gen}_n$ returns $(\sigma_0, \tilde{C}_0, \tilde{C}_1)$ and $(\sigma_1, \tilde{C}_0, \tilde{C}_1)$

**Eval$_n(i, (\sigma_i, \tilde{C}_0, \tilde{C}_1), (x', x''))$ is**

- $\tilde{s}_i\|\tilde{t}_i \leftarrow \text{Eval}_{n-1}(i, \sigma_i, x')$
- $\tilde{\rho}_i \leftarrow G'(\tilde{s}_i) \oplus \tilde{C}_{\tilde{t}_i}$
- return $\text{Eval}_{n-1}(i, \tilde{\rho}_i, x'')$
Distributed multi-point functions

- Just combine DPF-s
- Syntax: $f_{a_1,\ldots,a_n;b_1,\ldots,b_n}$
Lattice-based hardness assumption(s)

General set-up

- Let $G \xleftarrow{\$} \mathbb{F}^{k \times n}$ (where $k \ll n$), according to some distribution $G$
- Let $\vec{s} \xleftarrow{\$} \mathbb{F}^k$ (uniformly)
- Let $\vec{e} \xleftarrow{\$} \mathbb{F}^n$, according to some distribution $\mathbf{e}$
- Let $\vec{\eta} \xleftarrow{\$} \mathbb{F}^n$ (uniformly)

Assumption: $(G, \vec{s}^T \cdot G + \vec{e}^T) \approx (G, \vec{\eta}^T)$

- If $\mathbf{e}$ selects each component of $\vec{e}$ according to a Gaussian distribution centered at 0, then we have LWE
- If $\mathbf{e}$ selects a few random components of $\vec{e}$ uniformly randomly from $\mathbb{F}$, and the rest are 0, then we have LPN
Primal and dual LPN

- This was primal LPN. $G$ — generator matrix of a linear code
  - A way to break: guess a linearly dependent set of columns w/o errors, solve for $\vec{s}$
  - Works, if $n \in \Omega(k^2)$

Dual form of LPN

- Let $H \leftarrow \mathbb{F}^{n \times n'}$ (where $n' \leq n$)
- Let $\vec{e} \leftarrow \mathbb{F}^n$, according to low-Hamming-weight distribution
- Let $\vec{\gamma} \leftarrow \mathbb{F}^{n'}$ (uniformly)

Assumption: $(H, H \cdot \vec{e}) \approx (H, \vec{\gamma})$
Pseudorandom correlation generator (PCG) for R-VOLE

- **Setup**($x$) returns ($K_0$, $K_1$), where $x \in \mathbb{F}$ is a part of $K_1$
- **Expand**($0$, $K_0$) computes $\vec{u}$, $\vec{v}$; **Expand**($1$, $K_1$) computes $x$, $\vec{w}$, such that $\vec{v} + \vec{w} = \vec{u} \cdot x$
- **Setup** is to be run by a trusted dealer (later under 2PC)
- **Expand**($i$, $K_i$) is to be run by the $i$-th party
- Defining security for PCG requires care
Defining security of PCG-s

Let \((R_0, R_1)\) be distributed by the “real” construction. Let PCG expand \((K_0, K_1)\) into \((\tilde{R}_0, \tilde{R}_1)\)

Natural, but unrealizable way

\[(K_i, \tilde{R}_{1-i}) \approx (\text{Sim}(R_0), R_1)\]
Do not know, how to build Sim

Realizable way

\[(K_i, \tilde{R}_{1-i}) \approx (K_i, [R_{1-i} \mid R_i = \tilde{R}_i])\]
R-VOLE PCG based on Dual LPN

- Want R-VOLE of length $n$
- Construction parameters: $t, n' \in \mathbb{N}$, $t$ is “small”, $n' > n$
- There is a public matrix $H \in \mathbb{F}^{n' \times n}$

**Setup (input $x \in \mathbb{F}$)**

- Randomly pick $a_1, \ldots, a_t \in \{1, \ldots, n'\}$, $b_1, \ldots, b_t \in \mathbb{F}$
- Let $(K'_0, K'_1) \leftarrow \text{Gen}_{t\text{-DPF}}(f_{a_1, \ldots, a_t; b_1x, \ldots, b_tx} : \{1, \ldots, n'\} \rightarrow \mathbb{F})$
- Let $K_0 \leftarrow (K'_0, a_1, \ldots, a_t, b_1, \ldots, b_t)$
- Let $K_1 \leftarrow (K'_1, x)$
R-VOLE PCG based on Dual LPN

- Want R-VOLE of length \( n \)
- Construction parameters: \( t, n' \in \mathbb{N}, t \) is “small”, \( n' > n \)
- There is a public matrix \( H \in \mathbb{F}^{n' \times n} \)

Expand\((i, K_i)\)

- Let \( \vec{u}^b \in \mathbb{F}^{n'} \) have \( b_j \) at position \( a_j \)
- Let \( \vec{v}^b[i] \leftarrow \text{Eval}_{t\text{-DPF}}(0, K_0, i) \) for all \( i \in \{1, \ldots, n'\} \)
- Let \( \vec{w}^b[i] \leftarrow \text{Eval}_{t\text{-DPF}}(1, K_1, i) \) for all \( i \in \{1, \ldots, n'\} \)
- \( \vec{u} \leftarrow H \cdot \vec{u}^b, \vec{v} \leftarrow H \cdot \vec{v}^b, \vec{w} \leftarrow H \cdot \vec{w}^b \)
Evaluating Setup under 2PC

- Yes, it has to be evaluated under 2PC
- $\text{Gen}_{t,\text{DPF}}$ requires several invocations of the OWF
  - One round of AES (with already expanded key) is ca. 6800 AND-gates
- The rest is random number generation and $t$ multiplications in $\mathbb{F}$
Puncturable PRF-s (PPRF)

Keys $k$. Punctured keys $k\{\alpha\}$
GGM construction (PRG $\rightarrow$ PRF)

- Let $G : \{0, 1\}^\kappa \rightarrow \{0, 1\}^{2\kappa}$ be a PRG
- Let $\pi_0, \pi_1 : \{0, 1\}^{2\kappa} \rightarrow \{0, 1\}^\kappa$ take the first/last $\kappa$ bits of its argument

Construction of $F : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^\kappa$

(key and output lengths: $\kappa$. input length: $n$)

$$F_k(b_1 \cdots b_n) \leftarrow \pi_{b_n}(G(\cdots \pi_{b_2}(G(\pi_{b_1}(G(k)))) \cdots))$$

[Goldreich, Goldwasser, Micali. How to construct random functions. FOCS’84 / JACM’86]
Puncturing GGM construction

**Exercise.** Which values should be given to a party, such that he can compute \( F_k(x) \) for every \( x \in \{0,1\}^n \) but \( \alpha \)?
Oblivious PPRF

- Two parties. \( P_1 \) has \( k \). \( P_2 \) has \( \alpha \)
- They run a protocol. \( P_2 \) gets \( k\{\alpha\} \). \( P_1 \) gets nothing
- Protocol: one OT per layer
Active security for oblivious PPRF

- Expand the GGM tree one more time
- Let the left-hand leaves be the values of the PRF
- Use right-hand leaves to check correctness
  - compare linear combination
  - Include the XOR of all right-hand leaves in $k\{\alpha\}$
R-VOLE with 1-hot vector

- **Goal**: Sender gets \( \vec{u}, \vec{v} \) of length \( n \). Receiver gets \( x, \vec{w} \), s.t. \( \vec{u}x = \vec{v} + \vec{w} \) and \( \vec{u} \) is one-hot.

- Sender picks pos. \( \alpha \) and the value \( \beta = u[\alpha] \). Receiver picks \( x \).

- Receiver picks key \( k \) of PPRF. Expands it to vector of values \( \vec{w} \).

- Sender obliviously gets \( k\{\alpha\} \), expands it to \( \vec{v} \), except for \( v[\alpha] \).

- Run OLE, where
  - Receiver (as Sender) inputs \( x \) and \( r' \leftarrow F \)
  - Sender (as Receiver) inputs \( \beta \), obtains \( c \), s.t. \( r' + c = \beta x \)

- Receiver sends \( r \leftarrow r' - \sum_{i=1}^{n} w[i] \) to Sender

- Sender puts \( v[\alpha] \leftarrow c + r + \sum_{i \in \{1, ..., n\} \setminus \{\alpha\}} v[i] \)

- Do a consistency check that indeed, \( \vec{u}x = \vec{v} + \vec{w} \)
From 1-hot vectors to R-VOLE

- Want R-VOLE of length \( n \). Same parameters \( t, n', H \)

**Shared setup + Expansion (input \( x \in \mathbb{F} \))**

- Construct \( t \) VOLEs of length \( n \) with 1-hot vector:
  - Sender has \( \vec{u}_1, \ldots, \vec{u}_t, \vec{v}_1, \ldots, \vec{v}_t \)
  - Receiver has \( x = x_1 = \cdots = x_t, \vec{w}_1, \ldots, \vec{w}_t \)

- Let \( \vec{u}^b = \sum_{i=1}^{t} \vec{u}_i, \vec{v}^b = \sum_{i=1}^{t} \vec{v}_i, \vec{w}^b = \sum_{i=1}^{t} \vec{w}_i \)

- \( \vec{u} \leftarrow H \cdot \vec{u}^b, \vec{v} \leftarrow H \cdot \vec{v}^b, \vec{w} \leftarrow H \cdot \vec{w}^b \)
From VOLE to OT

- We’ll get $n$ Random-OT-s
- VOLE-Sender is OT-Receiver and vice versa
- Work over $\mathbb{F}_{2^\lambda}$, for e.g. $\lambda = 128$
- VOLE-Sender picks $\vec{u} \leftarrow \mathbb{F}_{2^n}$ and $\vec{v} \leftarrow \mathbb{F}_{2^\lambda}$
  - Not following the constraint on $\vec{u}$ can only hurt VOLE-Sender
- VOLE-Receiver picks $x \leftarrow \mathbb{F}_{2^\lambda}$
- Run VOLE. VOLE-Receiver gets $\vec{w} = \vec{u}x - \vec{v}$
- $i$-th OT:
  - OT-Receiver gets $u[i]$ and $\mathcal{H}(v[i])$
  - OT-Sender gets $\mathcal{H}(w[i])$ and $\mathcal{H}(w[i] \oplus x)$