Lecture 6: More on Distributed Point Functions
Plan

Recap: FSS & DPF Def'n
Application: PIR
DPF Construction
Stretch Break
Application: Private statistics

Logistics

- HW2 due Friday at 5pm via Gradescope
- OH on W 3-4:30pm
- Please ask & answer Qs on Piazza (use private Q if unsure)
Distributed Point Function (DPF)

"Way to succinctly share a structured vector."

\[
\text{Gen}(\alpha, \beta) \rightarrow \begin{array}{c}
\text{k}_0 \\
\text{k}_1
\end{array}
\]

\[
\begin{array}{c}
\text{Eval}(\text{k}_0) \\
\text{Eval}(\text{k}_1)
\end{array} \rightarrow \\
\begin{array}{c}
\text{long random-looking vector} + \text{(in } G) \\
\text{long random-looking vector}
\end{array}
\]

\[
\begin{array}{c}
0 \ 0 \ 0 \ \cdots \ \beta \ 0 \ \cdots \ 0 \ 0
\end{array}
\]

index \(\alpha\)

1. Correctness

\[
\forall \alpha \in \{0, 1\}^n, \ \beta \in G \ \forall i \in \{0, 1\}^n \ \forall (k_0, k_1) \leftarrow \text{Gen}(\alpha, \beta) \\
\text{Eval}(k_0) \cdot \text{Eval}(k_1) = \beta \cdot e_\alpha
\]

2. Security:

\[
\forall \alpha, \beta, \alpha', \beta' \\
\{ k_0 : (\alpha, \beta) \leftarrow \text{Gen}(\alpha, \beta) \} = \{ k_0 : (\alpha', \beta') \leftarrow \text{Gen}(\alpha, \beta) \}
\]

...same holds for \(k_1\).
Function secret sharing

Generalization of DIF to Fs.

DIF

\[ \begin{array}{c c c c c c c c c} 0 & 0 & 1 & 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 & 0 \end{array} \]

Fs for interval

\[ \begin{array}{c c c c c c c c c c} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & \cdots & \cdots & 0 & 0 \end{array} \]

Fs for general fn f

\( f(1), f(2), f(3), \cdots \) 

Don't have construction w/ short keys from DIF \( \Rightarrow \) unlikely
Simple PIR from DPFs

+ Very efficient, in both Comm & comp.

Claim 

1. If $\exists$ $t$-party DPF w/ key size $S(n)$, 
   $\exists$ $t$-server PIR w/ Comm $t \cdot S(n) + O(t)$

1. Correct since $a_0 + a_1 = \langle x, v_0 \rangle + \langle x, v_1 \rangle$

2. Secure by DPF $\langle x, v_0 + v_1 \rangle = \langle x, v_i \rangle = x_i$
Note: This DPF-based PIR scheme immediately gives a scheme for PIR by keywords.

$DB = \{s_1, \ldots, s_n\} \subseteq \{0,1\}^n$

$\forall i \in I$ 

$\sum_{i=1}^{n} Eval(k_o, s_i)$

$(k_o, k) \leftarrow \text{Gen(word, 1)}$
Constructing DPFs from OWFs.

Theorem: If PRGs exist, then for any security parameter $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, there is a two-party DPE construction with output space $\{0,1\}^\lambda$ with key size $O(\lambda n + n^2)$.

A slightly more clever construction can get rid of the $n^2$ term.

Disclaimer: This is my attempt at a very simple construction. Might be broken!

Proof: By induction on $n$.

Base case ($n = 0$):

When $n = 0$, DPF is just a secret-sharing scheme. Each share

\[
\text{Gen}_{\alpha, \beta} \left( \alpha, \beta \right) \rightarrow (k_0, k_i)
\]

Sample random $r_0, r_i \in \{0,1\}^\lambda$ s.t.

\[r_0 + r_i = \beta\]

Output $(r_0, r_i)$

\[\text{Eval}_{\alpha, \beta}(k) \rightarrow \text{output } k\]
**Induction Step** Proof by Picture

Use a PRG $G : \{0,1\}^n \rightarrow \{0,1\}^{2^n}$

$Gen_{n,\lambda}(\alpha, \beta) :$

Sample random seed $\alpha \in \{0,1\}^n$

$\alpha = \hat{\alpha} \| \alpha_n \in \{0,1\}^n \times \{0,1\}^n$

$(k_0, k_1) \leftarrow Gen_{n-1, \lambda+1}(\hat{\alpha}, \text{seed}\|1)$

---

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$\ldots$</th>
<th>$S_{2^n-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{2^n-1}$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CW**

---

$S_{2^n} + S'_{2^n}$  $S_{2^n} + S'_{2^n} + \beta$

---

Assuming $\alpha_n = 1.$
Almost the right thing. Just need this to be:

$$\begin{array}{c|c}
00000 & 00000 \\
00000 & 00000 \\
\vdots & \vdots \\
00000 & 00000 \\
\end{array}$$

$$\begin{array}{c|c}
\alpha & \beta \\
\end{array}$$
Induction Step: Assume for $n-1$.

Now we will construct a DPF on a domain of size $2^n$ from one on domain size $2^{n-1}$.

Use PRG $G: \{0,1\}^n \rightarrow \{0,1\}^{2^n}$

$$\text{Gen}_{n, \alpha, \beta}(\alpha, \beta)$$

Write $\alpha = 2 \parallel \alpha_n \in \{0,1\}^{n-1} \times \{0,1\}$

$\text{seed} \leftarrow \{0,1\}^n$.

$$(k_0, k_1) \leftarrow \text{Gen}_{n-1, \alpha_n, 1}(\alpha, \text{seed} \parallel 1)$$

$G(\text{seed}) \rightarrow (u_0, w_1) \in \{0,1\}^n \times \{0,1\}^3$

if $\alpha_n = 0$: $c_w \leftarrow [u_0 \oplus \beta, w_1]$

$\alpha_n = 1$: $c_w \leftarrow [u_0, w_1 \oplus \beta]$

$k_0 \leftarrow (k_0, c_w)$ $k_1 \leftarrow (k_1, c_w)$

output $(k_0, k_1)$
\[ \text{Eval}_{n,\lambda}(k, x) \]
\[
\text{parse } k = (\hat{k}, cu) \\
x = \hat{x} \parallel x_n \\
(S, b) \leftarrow \text{Eval}_{n-1, \lambda}(\hat{k}, x). \\
(w_0, w_1) \leftarrow G(S) \\
\text{if } b = 1: (w_0, w_1) \oplus = cw \\
\text{output } w_{x_n} \\
\]

**Correctness:** Follows by scrutinizing the construction.

**Security:** Follows by appealing to security of underlying DPF (induction), then to security of PRG.

**Efficiency:**
\[
S(n, 1) = \text{size of DPF key on domain } \{0,1\}^n \text{ with output bitlength } \lambda. \\
S(n, 1) \leq S(n-1, \lambda + 1) + 2^\lambda \\
\leq O(2^n + n^2). 
\]
Stretch
---
Break
Research Questions

* P-party DPF on n-bit domain w/ key size \( \text{poly}(p, 1, n) \)?

  \[ \leq \text{Best construction have size } O(1.2^n, 2^{n/2}) \]

* Can you improve the key size using a "simple" assumption (DDH)?

* Can you construct a FSS scheme from OWFs for the \( m \)-point function using less than DPF keys.

  \[ O(\lambda n m) \rightarrow O(\lambda n + m) \text{ bits per key?} \]
Application: Private Statistics

Each client (web browser) has a home page. Mozilla wants to know "Which home pages are most popular?"

Without learning any user's home page.

Can solve this type of problem with pretty good concrete efficiency using DPFs in the two-server setting.

Client $i$ has homepage $x_i \in \{0,1\}^n$

Two servers (one honest)

```
"nytimes.com is most popular."
```

Client $i$
Later on, we will see how to formalize security for these multiparty protocols.

Informally, want

1. **Correctness:** Everyone honest $\Rightarrow$ Servers get right answer.

2. **Security:** Adversary controls $\leq 1$ server and any # of clients $\Rightarrow$ Adversary learns nothing more than popular homepage (up to some leakage).
Warm-Up: Mozilla wants to know “How many clients have $\sigma \in \{0, 1\}^*$ as their home page?"

```latex
\begin{align*}
(Moz) & \quad \sum_{i=1}^{C} \text{Eval}(k_{i, \sigma}) \\
(MIT) & \quad \sum_{i=1}^{C} \text{Eval}(k_{i, \sigma})
\end{align*}
```

$\left( k_{o, i}, k_{i} \right) \leftarrow \text{Gen}(x_i, 1) \quad x_i \in \{0, 1\}^*$

---

Sum of server outputs

\[
\sum_{i=1}^{C} \text{Eval}(k_{i, \sigma}) + \sum_{i=1}^{C} \text{Eval}(k_{i, \sigma}) = \sum_{i=1}^{C} (\text{Eval}(k_{i,0}, \sigma) + \text{Eval}(k_{i,1}, \sigma)) = \sum_{i=0}^{C} (1 \text{ if } \sigma = x_i \text{ otherwise } 0) = \# \text{ clients holding string } \sigma
\]
More interesting case: Mozilla does not have a guess of popular URL in advance.

Idea: For each prefix length \( l \leq 1, \ldots, n \), run the known string protocol we just saw.

Mozilla asks a series of adaptive questions:

- How many clients have homepages starting with 0? With 1?
- ... 00? 01? 10? 11?

Prune search space when you encounter non-popular prefixes.

\[ \Rightarrow \] Can find all strings that > 1% of clients hold in time \( \# \) clients.

Catch: Leakage of prefix counts. Makes the security/leakage story a bit messy & unsatisfying.