Lecture 6: More on Distributed Point Functions

MIT - 6.893

Fall 2020

Henry Corrigan-Gibbs

Plan

Recap: FSS & DPF Def'n

Application: PIR

DPF Construction

Stretch Break

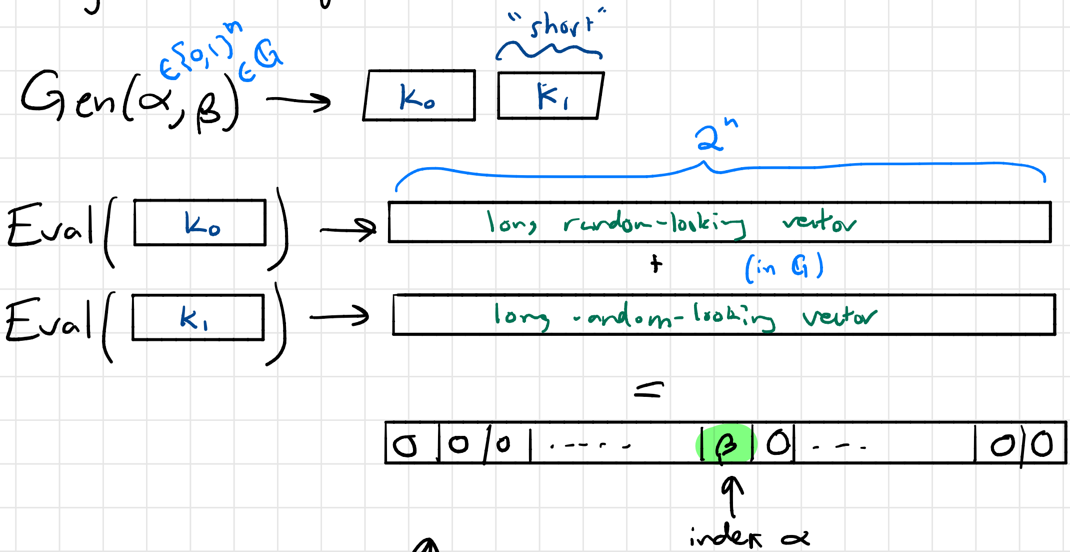
Application: Private statistics

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."



1. Correctness → holds

$\forall \alpha \in \{0,1\}^n, \beta \in \mathbb{G} \quad \forall i \in \{0,1\}^n \quad \forall (k_0, k_1) \leftarrow Gen(\alpha, \beta)$

$$Eval(k_0) + Eval(k_1) = \beta \cdot e_\alpha$$

2. Security: $\forall \alpha, \beta, \alpha', \beta'$

$$\{ k_0 : (\alpha, \beta) \leftarrow Gen(\alpha, \beta) \} \approx \{ k_0 : (\alpha', \beta') \leftarrow Gen(\alpha, \beta) \}$$

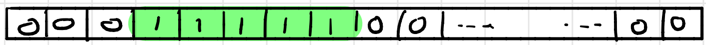
... same holds for k_1.

Function secret sharing

Generalization of DPF to fns.

DPF

FSS for interval



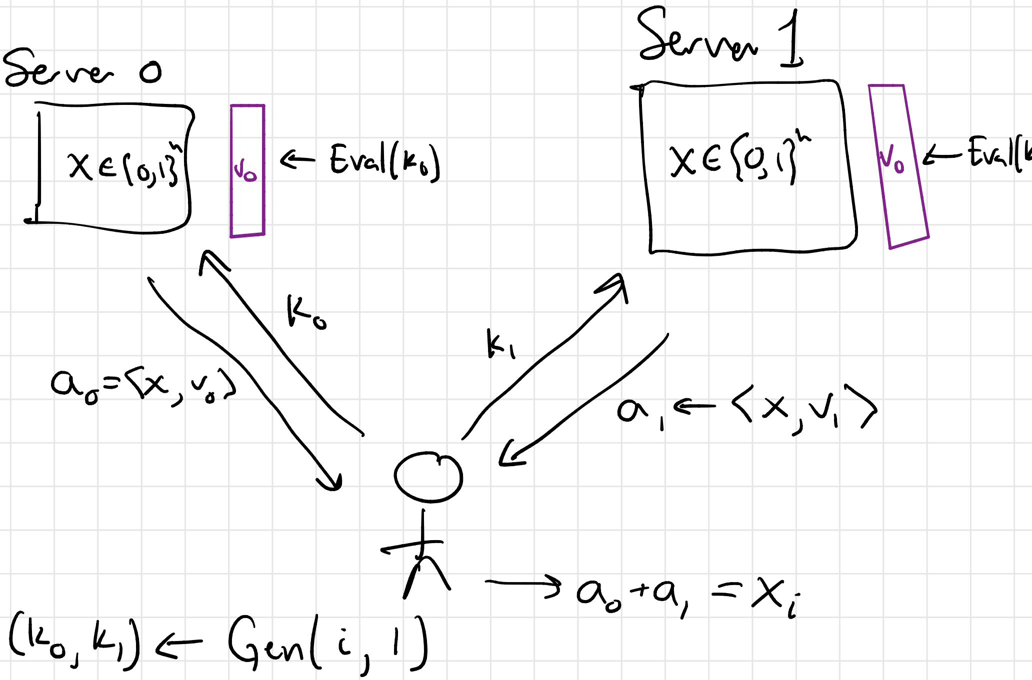
FSS for general fn f

don't have construction w/ short keys from OWF - unlikely

Simple PIR from DPFs

+ Very efficient, in both comm & comp.

Claim If ∃ t-party DPF w/ key size $S(n)$,
∃ t-server PIR w/ comm $t \cdot S(n) + O(t)$



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

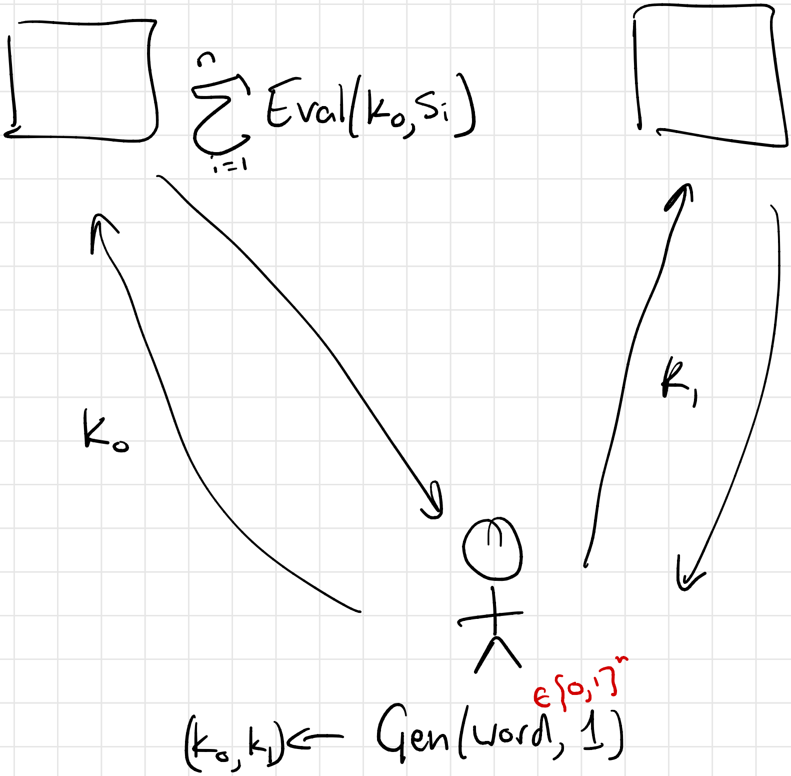
$= \langle x, v_0 + v_1 \rangle$

2. Secure by DPF security

$= \langle x, (0\ 0\ 0 \cdots 1 \cdots 0) \rangle = x_i$.

Note: This DPF-based PIR scheme immediately gives a scheme for PIR by keywords

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



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 DPF construction with output space $\{0,1\}^\lambda$ with key size $O(\lambda n + n^2)$

Exponential improvement over the naive scheme!

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

$Gen_{0,\lambda}(\alpha, \beta) \to (k_0, k_1)$ where $\alpha \in \{0,1\}^0 = \perp$, $\beta \in \{0,1\}^\lambda$

- Sample random $r_0, r_1 \in \{0,1\}^\lambda$ s.t. $r_0 + r_1 = \beta$
- Output (r_0, r_1)

$Eval_{0,\lambda}(k) \to$ output k.

Induction Step — Proof by Picture

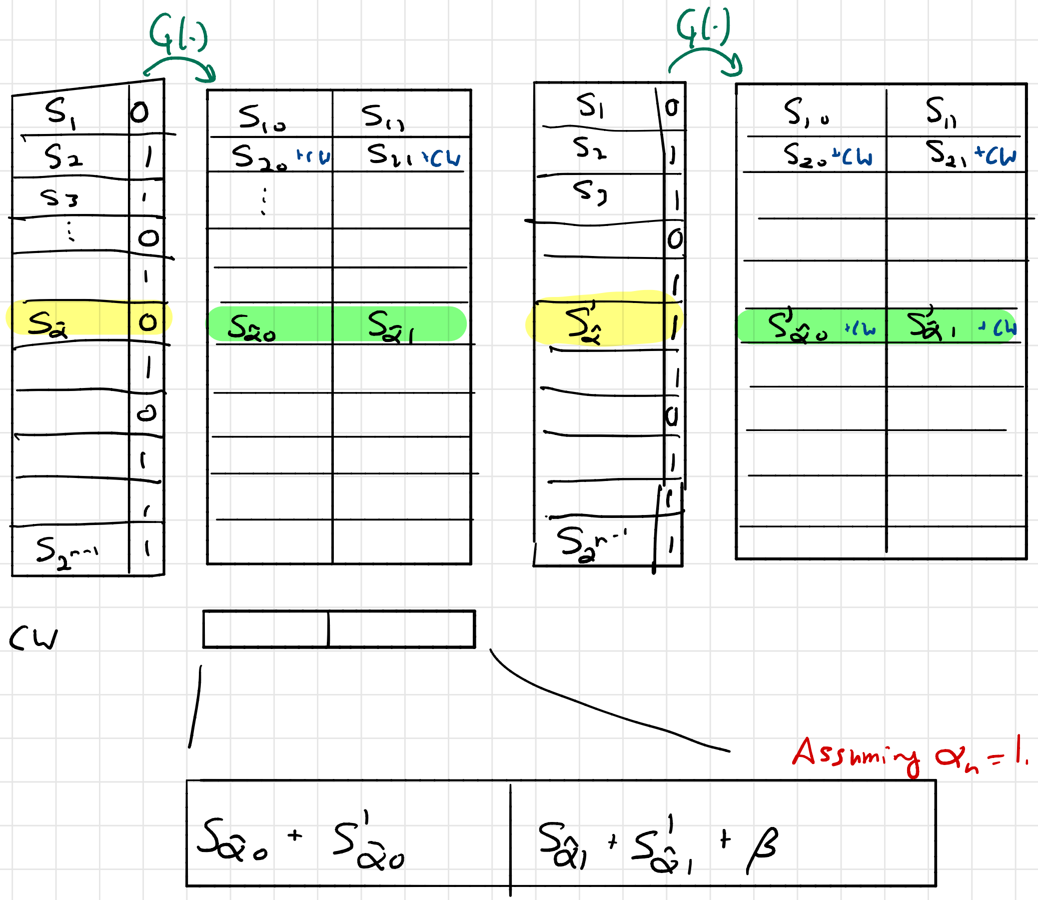
Use a PRG $G: \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$.

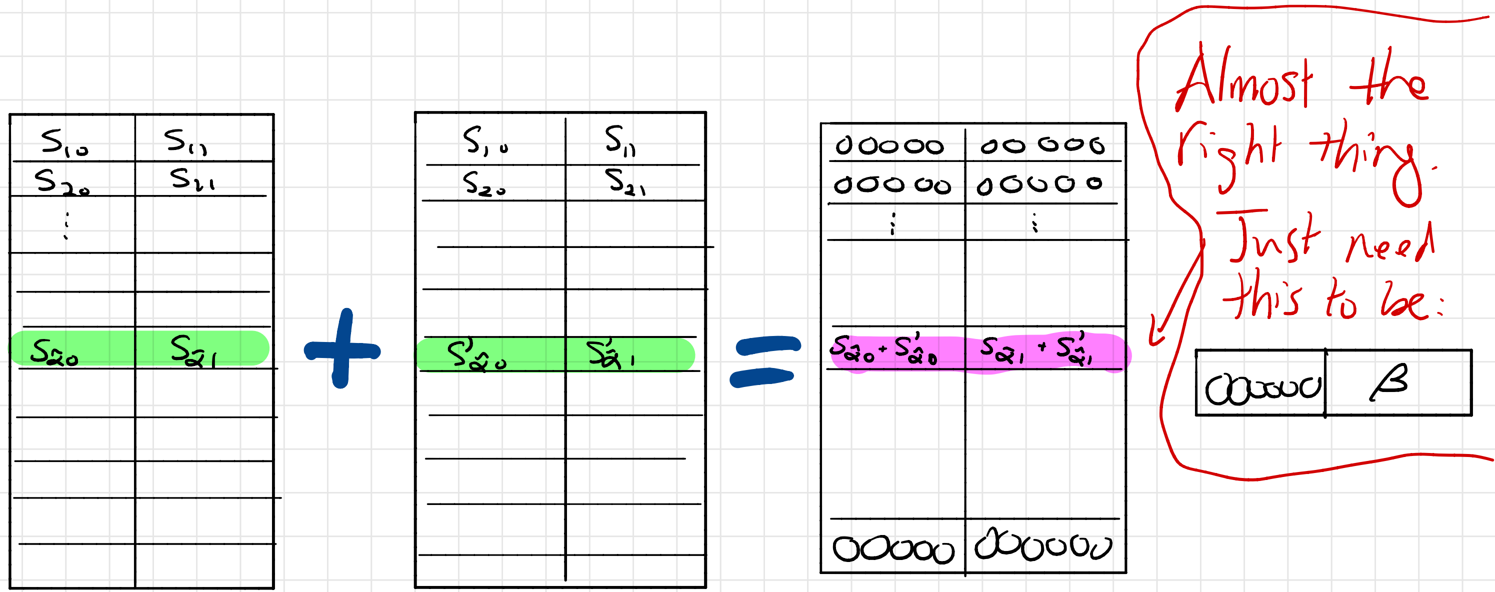
$\text{Gen}_{n,\lambda}(\alpha, \beta)$:

Sample random seed $\leftarrow^{\$} \{0,1\}^\lambda$

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

$(\hat{k}_0, \hat{k}_1) \leftarrow \text{Gen}_{n-1,\lambda+1}(\hat{\alpha}, \text{seed} \| 1)$





Almost the right thing.
Just need this to be:

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 $G: \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda}$

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

- Write $\alpha = \hat{\alpha} \| \alpha_n \in \{0,1\}^{n-1} \times \{0,1\}$
- $\mathrm{seed} \xleftarrow{R} \{0,1\}^{\lambda}$.
- $(\hat{k}_0, \hat{k}_1) \leftarrow \mathrm{Gen}_{n-1,\lambda+1}(\hat{\alpha}, \mathrm{seed} \| 1)$
- $G(\mathrm{seed}) \rightarrow (w_0, w_1) \in \{0,1\}^{\lambda} \times \{0,1\}^{\lambda}$
- if $\alpha_n = 0$: $\quad cw \leftarrow [w_0 \oplus \beta, w_1]$
- $\quad \alpha_n = 1$: $\quad cw \leftarrow [w_0, w_1 \oplus \beta]$
- $k_0 \leftarrow (\hat{k}_0, cw) \qquad k_1 \leftarrow (\hat{k}_1, cw)$
- output (k_0, k_1)

$\mathrm{Eval}_{n,\lambda}(k, x)$

parse $k = (\hat{k}, cw)$
$x = \hat{x} \| x_n$

$(s, b) \leftarrow \mathrm{Eval}_{n-1, \lambda+1}(\hat{k}, \hat{x})$.

$(w_0, w_1) \leftarrow G(s)$

if $b = 1$: $(w_0, w_1) \oplus= cw$

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: $S(n, \lambda)$ = size of DPF key on domain $\{0,1\}^n$ with output bitlength λ.

$$S(n, \lambda) \leq \underbrace{S(n-1, \lambda+1)}_{\hat{key}} + \underbrace{2\lambda}_{cw}$$

$$\leq O(\lambda n + n^2).$$

Stretch

Break

Research Questions

* P-party DPF on n-bit domain w/ key size poly(p, λ, n)?
  - ↳ Best constructions have size $O(\lambda \cdot 2^{p/2} 2^{n/2})$
* Can you improve the key size using a "simple" assumption (DDH)?

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

$O(\lambda n m) \longrightarrow O(\lambda n + m)$ bits per key?

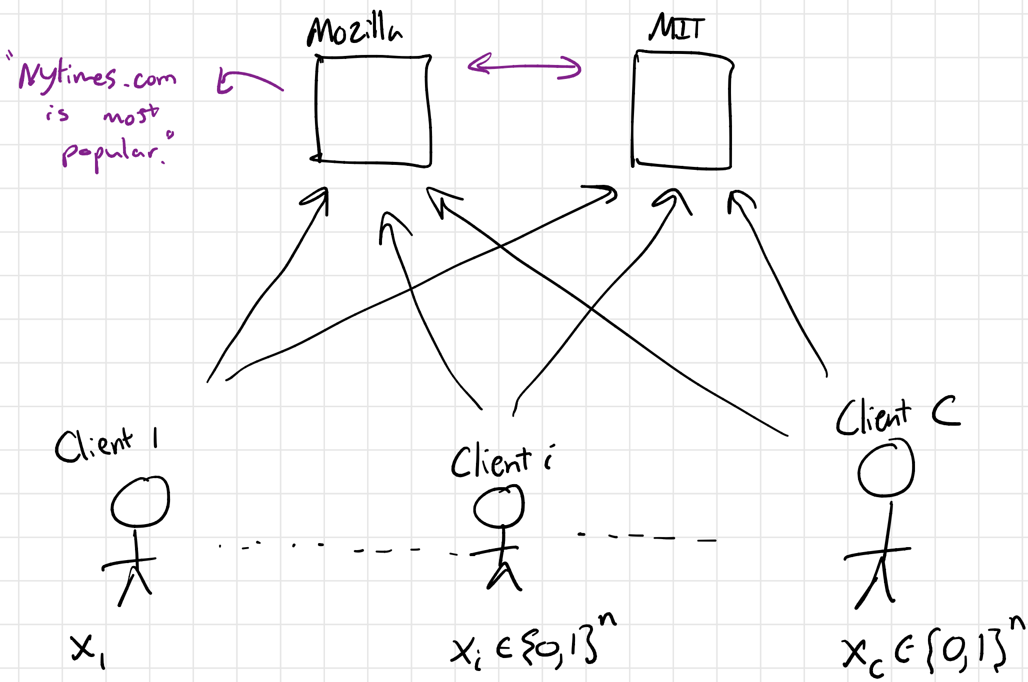
Application: Private Statistics

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 problems with pretty good concrete efficiency using DPFs in the two-server setting.

Client i has home page $x_i \in \{0,1\}^n$
Two servers (one honest)



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

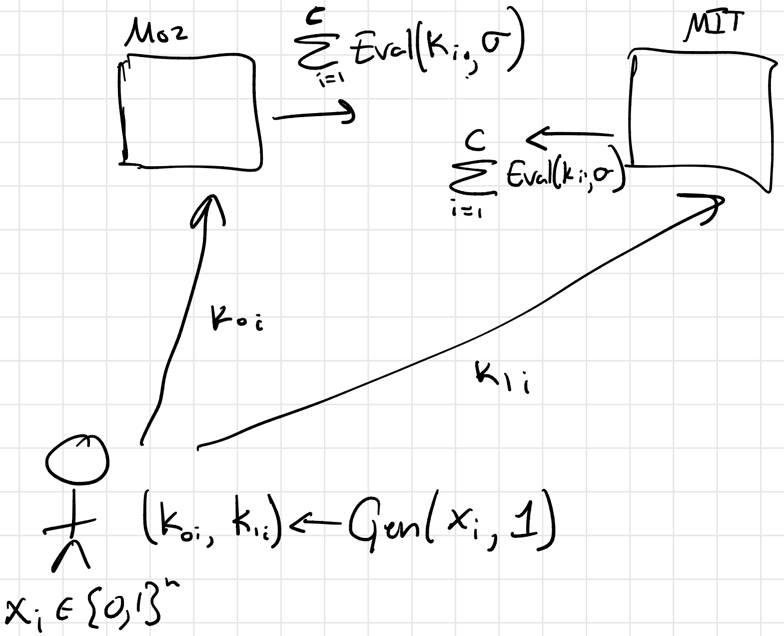
Informally, want

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

2. Security: Adv controls ≤ 1 server and any # of clients $\Rightarrow$ Adv learns nothing more than popular homepage (up to some leakage)

Warm Up: Mozilla wants to know

"How many clients have $\sigma \in \{0,1\}^n$ as their home page?"



Sum of server outputs

$$\sum_{i=1}^{c} Eval(k_{i_0}, \sigma) + \sum_{i=1}^{c} Eval(k_{i_1}, \sigma)$$

$$= \sum_{i=1}^{c} \left(Eval(k_{i_0}, \sigma) + Eval(k_{i_1}, \sigma) \right)$$

$$= \sum_{i=0}^{c} \begin{pmatrix} 1 \text{ if } \sigma = x_i \\ 0 \quad \text{o.w} \end{pmatrix}$$

$= \#$ clients holding string σ

More interesting case: Mozilla does not have a guess of popular URL in advance.

Idea: For each prefix length $\ell \in \{1, \dots, n\}$, 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?
with 00? 01? 10? 11?

⋮

Prune search space when you encounter non-popular prefixes.

⇒ Can find all strings that > 1% of clients hold in time linear in # clients.

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