This class: "secure computation on secret data"

So far we've seen FHE / lattice techniques

FHE: $Enc_{pk}(x) \xrightarrow[Dec]{Eval} f(x)$

No circuit-dependent comm.

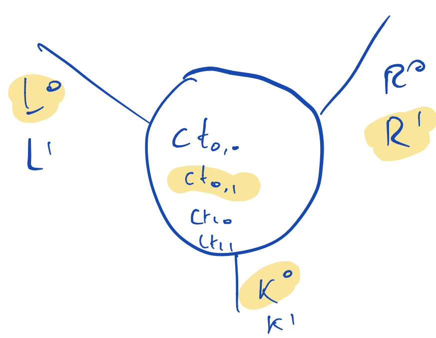
$\Rightarrow$ Narrow band of assumptions (lattice based)

Conceptually: eggs in one basket

Practical: heavy, slow

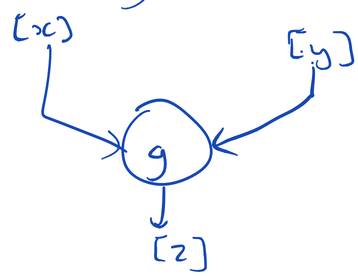
$\Rightarrow$ not true for GMW/GC

Garbled Circuits



General idea: Progress through circuit gate ↳ gate obtaining wire labels

Secret-sharing based: GMW/BGW



$[x]: x_a + x_b = x$

val idea: progress through circuit gate by gate maintaining this invariant

Compare & Contrast

FHE	GC/GMW/BGW
succinct comm.	Comm. per gate
Narrow band of assumptions (lattices) - Conceptually, eggs in one basket - Practical: powerful crypto runs slower	Generically instantiable (OT, OWFs) · Fast in practice with lower grade crypto (eg. AES, elliptic curves)

HSS: secret-sharing analogue of FHE

Benaloh 86 ⇒ distributed

Boyle Gilboa Ishai-15 ⇒ non-interactive evaluation

$$\begin{tikzcd}
 & s_0 \arrow[r, "\text{Eval}"] & f_0(s_0) \arrow[dr] & \\
s \arrow[ur] \arrow[dr] & & & f(s) \\
 & s_1 \arrow[r, "\text{Eval}"] & f_1(s_1) \arrow[ur, "\text{Recon}"] &
\end{tikzcd}$$

Can we realize this with a broader class of assumptions than FHE?

i.e. with weaker, faster primitives while also maintaining meaningful efficiency

It's been a few years:

Ishai's Worlds of HSS (a la Impagliazzo)

Algo. < Minicrypt < Lapland < Cryptomania < Gentria

Extremes:

Trivial

P = NP

Algorithmica: Linear functions

[Benaloh, GMW]

$$
\begin{tikzcd}
 & x_0 \arrow[r, "f"] & ax_0+b \arrow[dr] & \\
x \arrow[ur] \arrow[dr] & + & + & ax+b \\
 & x_1 \arrow[r, "f"'] & ax_1 \arrow[ur] &
\end{tikzcd}
$$

$x_0 + x_1 = x$

High-end:

"Gentria": LWE+

Circuits, from Spooky encryption

Interesting stuff is what lies in between

Cryptomania:

Branching programs from specific structured assumptions
ie.. DDH, Paillier, LWE
(not generic PKE / OT)

DDH construction is Eysons talk

Laplond : Constant degree multivariate polynomials high noise
Pseudorandom Correlation Generators, to
Expand
OT, OLE, etc. from LPN parameters
not known to give PKE. Exciting MPC work

Minicrypt : One-way functions

⇒ Point functions ++ (intervals, decision trees)

Here the results & exposition framed better as the dual notion of Function Secret Sharing

Function Secret Sharing (FSS)

Recall HSS:

$$\begin{tikzcd} & x_0 \arrow[r, "\mathrm{Eval}(f)"] & y_0 \arrow[dr] & \\ x \arrow[ur] \arrow[dr] & & & y = f(x) \arrow[l, phantom, "\mathrm{Dec}"] \\ & x_1 \arrow[r, "\mathrm{Eval}(f)"] & y_1 \arrow[ur] & \end{tikzcd}$$

FSS:

$$\begin{tikzcd} & f_0 \arrow[r, "\mathrm{Eval}(x)"] & y_0 \arrow[dr] & \\ f \arrow[ur] \arrow[dr] & & & y = f(x) \arrow[l, phantom, "\mathrm{Dec}"] \\ & f_1 \arrow[r, "\mathrm{Eval}(x)"] & y_1 \arrow[ur] & \end{tikzcd}$$

Parameters: $p \in \mathbb{N}$ parties, function class $\mathcal{F}$, secparam λ

Gen: $1^\lambda, f \longmapsto k_1, k_2 \ldots k_p$ (p-parties)

Eval: $i, k_i, x \longmapsto y_i$

Dec: $y_1, y_2 \ldots y_p \longmapsto y$

Correctness: $\forall f \in \mathcal{F}$, $x \in \text{Domain}(f)$

$\forall k_1, \ldots k_p \leftarrow \text{Gen}(1^\lambda, f)$,

$\text{Dec}(\text{Eval}(1, k_1, x) \ldots \text{Eval}(p, k_p, x)) = f(x)$

Security: m-party, t-secure FSS: for $\mathcal{F}$

$\forall S \subset [m]$ s.t. $|S| = t$, $\exists$ PPT Sim

s.t. $\forall f \in \mathcal{F}$, the following are indist.:

$\text{Real}(1^\lambda)$: $k_1, \ldots k_m \leftarrow \text{Gen}(1^\lambda, f)$

output $(k_i)_{i \in S}$

$\text{Ideal}(1^\lambda)$: output $\text{Sim}(1^\lambda, S)$

can include leakage

$\text{Real}_A \approx_c \text{Ideal}_A$

Let's rule out

Unwanted constructions:

Gen(f): interpret f as bit-string $\in \{0,1\}^{|f|}$

Sample $f_0, f_1 \leftarrow \{0,1\}^{|f|}$

$f_0 \oplus f_1 = f$

output f_0, f_1

Eval(i, f_i, x): output $f_i, x \Rightarrow y_i$

Dec(y_i): reconstruct $f = f_0 \oplus f_1$

output $f(x)$

"Function privacy"? from Dec

Gen(f): $\tilde{C}, (X_{i0}, X_{i1})_{i \in [n]} \leftarrow Gb(f)$

output $f_0 = \tilde{C}$, $f_1 = (X_{i0}, X_{i1})_{i \in [n]}$

Eval(i, f_i, x): if $i = 0$ output $\tilde{C}$

else output $(X_{j x_j})_{j \in [n]}$

$Dec(y_0, y_1)$: output $Ev(\tilde{C}, \tilde{x})$

Note: works for one evaluation, for many evals need to rerandomize $\tilde{C}, \tilde{x}$

Unsatisfying: output shares are massive, real work done by Dec

Therefore: Linear decoding procedure: $Dec: \sum y_i$

Not formal $\in$
- succinctness: share output size comparable to plain output
- Compressibility

Properties become clear with an application

Specifically constructing FSS for class of point functions, aka Distributed Point Functions

Point function:

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

FSS for point functions: elegant soln. to PIR
(aka DPF)

Private Information Retrieval

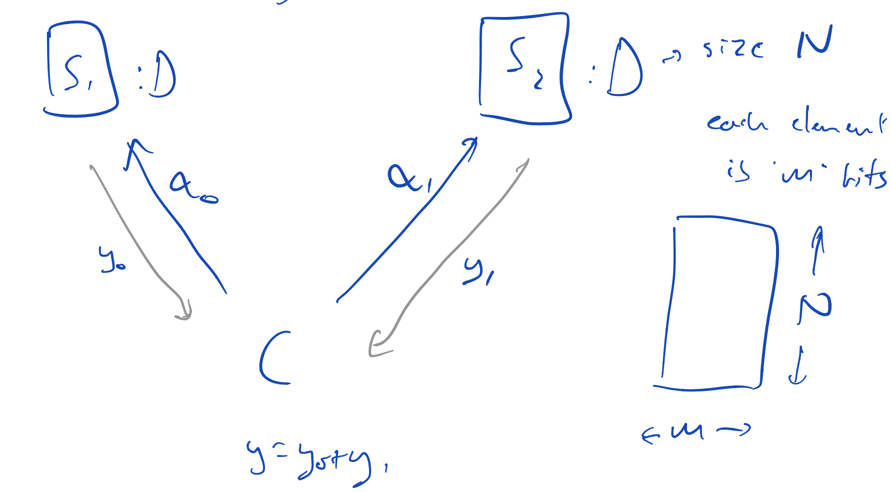
- 'Public' database D:
- Client 'C' wishes to read entry at location 'α'
  without revealing 'α' to database holder

We wish to achieve this without sending
D to client

Long history, someone else will cover

2-Server case

Non-colluding servers



Solution with FSS for point functions

C: ① define point function $f_{\alpha,1}$

i.e. $f_{\alpha,1}(x) = \begin{cases} 1 & \text{if } x = \alpha \\ 0 & \text{otherwise}, \; x \in [N] \setminus \alpha \end{cases}$

Note: domain of 'f' is size of database N

② $k_1, k_2 \leftarrow Gen(f_{\alpha,1})$

③ Send k_1 to S_1, k_2 to S_2

S_i : $y_i = \sum_{j \in [N]} x_j \cdot \mathrm{Eval}(i, k_i, j)$

· Send y_i to C

$\downarrow$

just m bits
(same size as actual output)

C : output $x = y_0 + y_1$

<u>Correctness</u>:

$$
\begin{aligned}
x &= y_0 + y_1 \\
&= \Big(\sum_j x_j \cdot \mathrm{Eval}(0, k_0, j)\Big) + \Big(\sum_j x_j \cdot \mathrm{Eval}(1, k_1, j)\Big) \\
&= \sum_{j \in [N]} x_j \cdot \underbrace{\big(\mathrm{Eval}(0, k_0, j) + \mathrm{Eval}(1, k_1, j)\big)}_{f_{\alpha,1}(j)}
\end{aligned}
$$

$\Rightarrow$ correctness of FSS

$$= \sum_{j \in [N]} x_j \cdot f_{\alpha,1}(j)$$

$$\begin{cases} 1 & \text{if } j = \alpha \\ 0 & \text{otherwise} \end{cases}$$

only nonzero value in the sum is $j = \alpha$

$$= x_\alpha \cdot 1$$

Security: each S_i only gets one share of $f_{\alpha,1}$ $\Rightarrow$ servers know $\mathcal{F}$ (i.e. that it is a pt. fn.) but they don't know which point $\Rightarrow$ no info abt α

Technical: view of each server is simulatable, simply run $\mathrm{Sim}(1^\lambda)$ of FSS scheme

View server i = $\mathrm{Sim}(1^\lambda, i)$

Efficiency:

Computation for C: 1 FSS Gen

Communication complexity

- C transmits k_i to S_i
- S_i sends y_i to C

y_i → same size as x_α

Key size matters!

Ruling out another trivial instantiation:

additively share the entire truth table of f

i.e. $K_1 = (K_{11}, K_{12} \dots K_{1N})$

$K_2 = (K_{21} \; K_{22} \dots K_{2N})$

such that $K_{1i} \oplus K_{2i} = f(i)$

client work and comm. linear in N

$\Rightarrow$ not interesting

Interesting from this line of work:

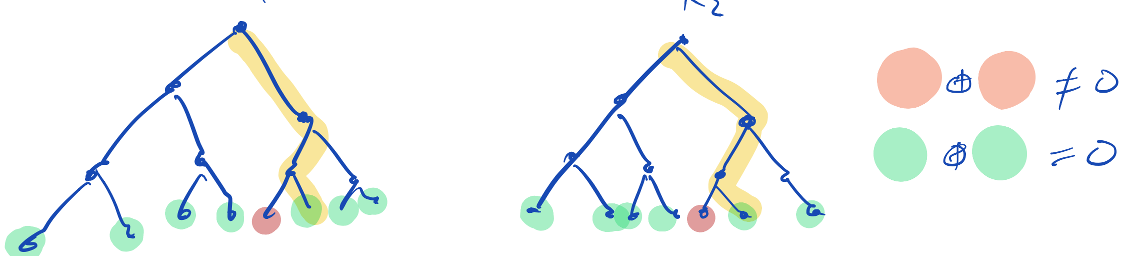
Thm. Assuming OWFs, there is a two key FSS scheme for the family of point functions with key size $O(\lambda \cdot \log N)$

High level idea:

K_1, K_2 will define GGM style binary tree with N leaves where each leaf is an 'evaluation point'

assuming a length doubling PRG:

K_1 $\quad x = 101 \quad$ K_2



In GGM: trees fully specified by K

here: bit much to ask, we also supply some kind of helper "correction word" for each level of the tree.

Building blocks: weak homomorphism + conditional correction

Idea ① 'weak homomorphism' from PRG + additive secret sharing

Let $[s] = (s_1, s_2) : s_1 \oplus s_2 = s$

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

Define $G([s]) = (G(s_1), G(s_2)) = [\sigma]$

abuse of notation

$G(s_1) \oplus G(s_2)$

Two cases:

1) if $s=0$, $s_1 = s_2$

$G([0]) = (G(s_1), G(s_1)) = [0]$

2) else $s \neq 0$, $s_1 \neq s_2$

$G([s]) = (G(s_1), G(s_2)) \Rightarrow$ (pseudo) random

$= [\sigma]$

$\rightsquigarrow G(s_1) \oplus G(s_2)$

Takeaway: $G([s])$ expands small $[0] \to [0]$

small $[\text{rand}] \to [\text{rand}]$

Idea ②: Conditional correction

Let $[s] = (s_1, s_2)$, $s_1 \oplus s_2 = s \in \{0,1\}^{k}$

$[t] = (t_1, t_2)$, $t_1 \oplus t_2 = t \in \{0,1\}$

"control bit"

$c \in \{0,1\}^{k}$: public correction word

Locally computable :

$$[s \oplus t \cdot c] = (s_1 \oplus t_1 \cdot c,\ s_2 \oplus t_2 \cdot c)$$

sanity check: $s_1 \oplus t_1 c \oplus s_2 \oplus t_2 c$

$$= (s_1 \oplus s_2) \oplus (t_1 \oplus t_2) c$$

$$= s \oplus t \cdot c$$

Syntax: $[s], [t]$

c

Expand to child nodes:

apply weak homomorphic cond'l. correction

L $\qquad\qquad\qquad\qquad$ R

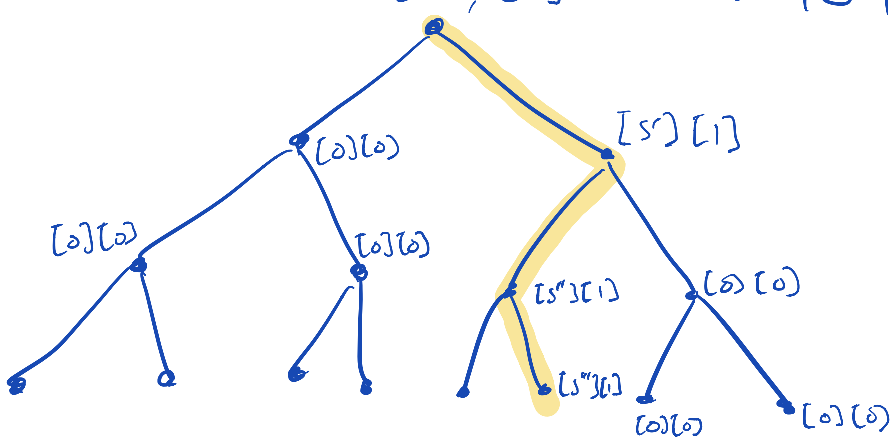
$[s_L], [t_L]$ $\qquad\qquad\qquad\qquad$ $[s_R], [t_R]$

$$[s_L \quad t_L \,, \quad s_R \,, \quad t_R] = a([s]) \oplus t \cdot c = [\sigma \oplus t \cdot c]$$

Invariant: $[s][t] = \begin{cases} [0][0] & \text{on special path} \\ [\text{rand}][1] & \text{off special path} \end{cases}$

Composition

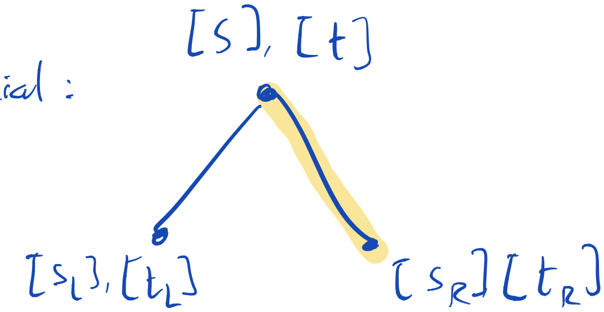
$[s], [t]$ $\qquad$ $a = 101$



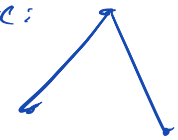
Two types of parent → children expansion

Type 1:

Special:
↓
root



Inactive:



Recall $[s] = (s_1, s_2)$ $s_1 \oplus s_2 = s$

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

$G([s]) = (G(s_1), G(s_2)) = [\sigma]$

$\sigma \in \{0,1\}^{2\lambda+2}, \sigma = G(s_1) \oplus G(s_2)$

Parse $\sigma_L \| \sigma_R = \sigma$

$\sigma_L, \sigma_R \in \{0,1\}^{\lambda+1}$

Our goal: set $\overset{\text{correction word}}{C}$ such that

Inactive $[s_L][t_L] = [0][0]$

Active $[s_R][t_R] = [\text{rand}][1]$

Parse $C = C_L, C_R \in \{0,1\}^{2(\lambda+1)}$

$[s_L, t_L] = [\sigma_L \oplus t \cdot C_L]$

$[0^\lambda, 0] = [\sigma_L \oplus C_L]$

$\therefore C_L = \sigma_L$

What about C_R?

$[s_R, t_R] = [\sigma_R \oplus t \cdot C_R]$

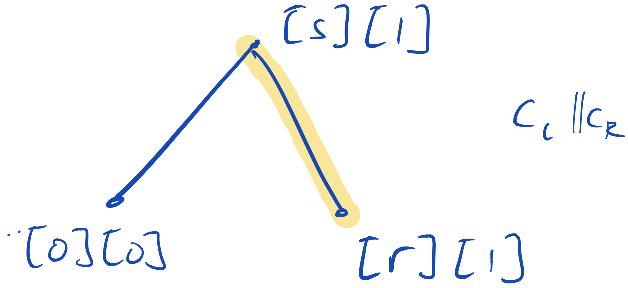
$r \in \{0,1\}^\lambda$

$[r, 1] = [\sigma_R \oplus C_R]$

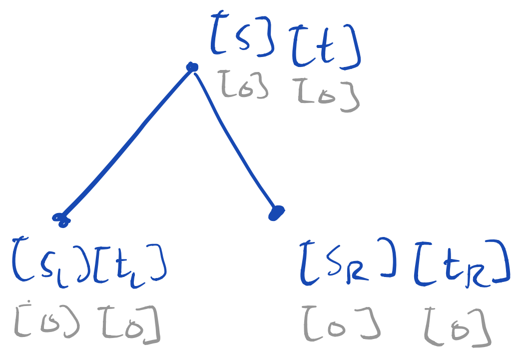
$\therefore C_R = \sigma_R \oplus (r \| 1)$

Note: for a leaf node, $r = \beta$

Correctness of evaluation



Type 2 (plain)



c

Eval.: Same as earlier
(agnostic to on vs. off special)

$$[s_L, t_L, s_R, t_R] = [\sigma \oplus t \cdot c]$$

Remember expansion is zero-preserving

$\therefore \sigma = C_1([s]) = C_1([0]) = [0]$

Also $[t] = [0]$

$\therefore [\sigma \oplus t \cdot c] = [0 \oplus 0] = [0]$

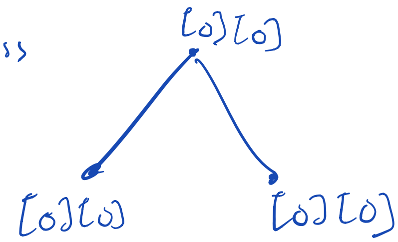
Independent of correction word

$\therefore$

$$[s_L, t_L, s_R, t_R] = [\sigma \oplus t \cdot c] = [0,0,0,0]$$

$\Rightarrow$ Invariant preserved

Correctness



Putting it together:

Gen: $f_{\alpha,\beta}$, say $|\alpha| = 3$ bits

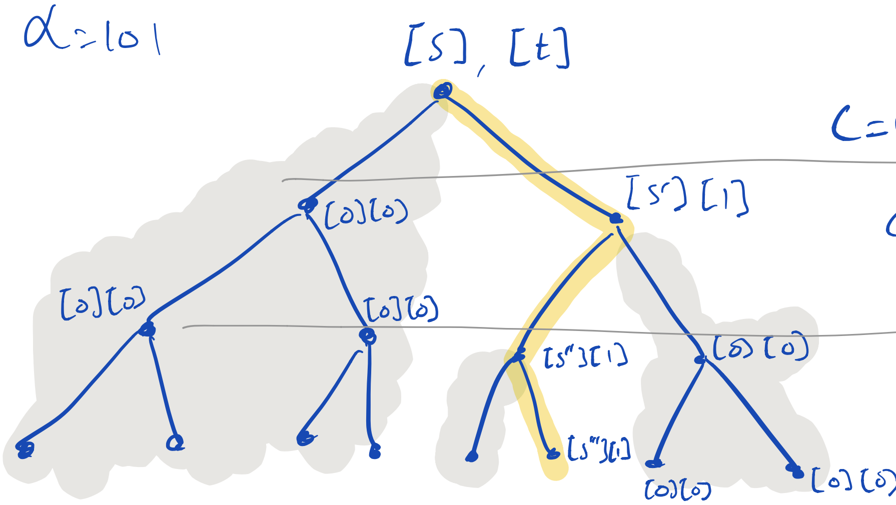
$[s] = (s_1, s_2) \leftarrow \{0,1\}^{2\lambda}$, $[t] \leftarrow \{0,1\}^2$

$[s'] =$ "

$[s''] =$ "

$[s'''] = [\beta]$

$\alpha = 101$



$C = \sigma_{L'} (\sigma_R \oplus s' \| 1)$

$C' = \sigma_L' \oplus s'' \| 1, \sigma_R$

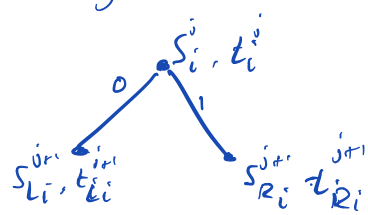
$C'' = \sigma_L, \sigma_R \oplus \beta \| 1$

Output keys $k_1 = s_1, C, C', C''$

$k_2 = s_2, C, C', C''$

<u>Eval</u>: $i, k_i, x = x_0 x_1 x_2$

At layer $j \in |x| = \{0,1,2\}$:



$C^j \in \{0,1\}^{2\lambda+2}$

$s_{Li}^{j+1}, t_{Li}^{j+1}, s_{Ri}^{j+1}, t_{Ri}^{j+1} = G(s_i^j) \oplus t_i^j \cdot C$

if $x_j = 0$, set $s_i^{j+1}\ t_i^{j+1} = s_{Li}^{j+1}\ t_{Li}^{j+1}$

else set $s_i^{j+1}\ t_i^{j+1} = s_{Ri}^{j+1}\ t_{Ri}^{j+1}$

Security :

$K_1 = s_1, t_1, c, c', c''$

$H_0 \quad = s_1, t_1, \sigma_L \| \sigma_R \oplus s', \sigma_L' \oplus s'' \| \sigma_R, \sigma_L \| \sigma_R \oplus \beta$

Hybrid 0 : $\sigma_L \sigma_R = G(\{s\}) = G(s_1) \oplus G(s_2)$

Hybrid 1: $\sigma_L \sigma_R \leftarrow \{0,1\}^{2\lambda+2}$

$H_0 \approx_c H_1$ by PRG security (s_2 not in view) & s', s'', β independent of s

$H_1 : \sigma_L' \sigma_R' = G(s_1') \oplus G(s_2')$

Hybrid 2: $\sigma_L' \sigma_R' \leftarrow \{0,1\}^{2\lambda+2}$

Hybrid 3: $\sigma_L'', \sigma_R'' \leftarrow \{0,1\}^{2\lambda+2}$

Hybrid 4 (Simulation): s_1, t_1, c, c', c'' all sampled uniformly

Extension to intervals: simply add FSS for multiple points. Tweaks: comparison

Extending to many parties:

Thm. assuming OWFs, $\exists$ p-key FSS for point functions with key length $O(2^{n/2} \cdot 2^{p/2} \cdot m)$

Assuming p is const., better than trivial $(O(2^n \cdot m))$

Bonus: Silent OT

PCG: Pseudorandom Correlation Generator

Correlation Generator: GenCor: c_1, c_2

e.g. Beaver triples.

Gen: s_1, s_2

$c_i = x_i, y_i, z_i$
$(x_1+x_2)(y_1+y_2) = z_1+z_2$

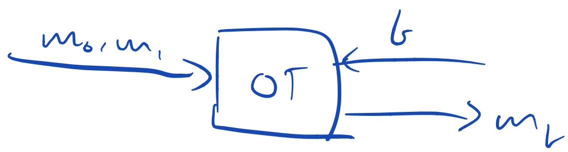
Expand (s_i): c_i

$c_1, c_2 \approx \text{GenCor}(1^\lambda)$

technical: reverse sampling

Oblivious Transfer:

S: m_0, m_1 R: b



But: OT needs public key operations

Impagliazzo-Rudich ...

$\Rightarrow$ expensive

Beaver 96, IKNP : few OTs
shai ilian issim etrank

"stretch" to many OT instances via symmetric key crypto

↓ OWF

↓ CorRob HF

⇓

S: $\Delta \in \{0,1\}^\lambda$, $\vec{r} \in \{0,1\}^{\lambda \times m}$

R: $\vec{b} \in \{0,1\}^m$ choice bits, $\vec{q} \in \{0,1\}^{\lambda \times m}$

OT Correlation: $q_i \oplus r_i = \Delta \cdot b_i$

How do we use this?

$H(r_i)$ serve as OTPs to

$H(r_i \oplus \Delta)$ encrypt i^{th} message pair

Cor.-robust: $H(r \oplus \Delta)$ appears random

R decrypts using $H(r_i \oplus b_i \Delta) = H(q_i)$

Task: compress this correlation into

short seeds s_1, s_2

Thm: Assuming LPN, there is a

2key PCG for OT correlations

For some integers m, n, t

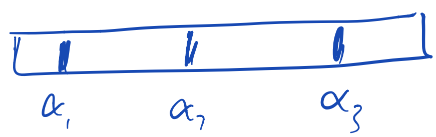
$\vec{\alpha}$ $\alpha_1, \alpha_2 \ldots \alpha_t \in [m]$

Define multi point function $f_{\vec{\alpha}, y}(x) = \begin{cases} \Delta & \text{if } x \in \vec{\alpha} \\ 0 & \text{otherwise} \end{cases}$

FSS for multi-point fns (lin. comb. of p.f.)
produce keys K_1, K_2

So on $S_1 : K_1, \vec{\alpha}$

$S_2 : K_2, \Delta$

Define $\vec{b_i} =$ 

$b_i = 1$ if $i \in \vec{\alpha}$ $\Rightarrow$ fully specified by S_1
0 otherwise

Define $q_i = \text{Eval}(0, K_1, i)$
$r_i = \text{Eval}(1, K_2, i)$

$$q_i \oplus r_i = b_i \cdot \Delta$$

Looks like OT correlation, but we're not there yet.

This is where LPN comes in

Dual LPN assumption:

parity check matrix

$$\underbrace{\Big[\quad H \quad\Big]}_{\longleftarrow\; m \;\longrightarrow}^{\updownarrow n} \cdot \begin{bmatrix} e \end{bmatrix} \approx_c \begin{bmatrix} u \end{bmatrix} \updownarrow n$$

binary vector of low HW sampled according to some distr.

H is public, so matrix mult is a linear operation on the shares

Expand $(s_1 = k_1, \vec{\alpha})$	$E\,(s_2 = k_2, \Delta)$
$H \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix} = \begin{bmatrix} q_1' \\ \vdots \\ q_n' \end{bmatrix}$	$H \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix} = \begin{bmatrix} r_1' \\ \vdots \\ r_n' \end{bmatrix}$

$$q_i' = \langle H_i, q_i \rangle$$

$$r_i' = \langle H_i, r_i \rangle = \langle H_i, q_i \oplus b_i \Delta \rangle = q_i' \oplus \Delta \langle H_i, b_i \rangle$$

$= q_i \oplus \Delta \cdot b_i$

By dual-LPN, $\vec{b}$ is pseudorandom & specified by K_1

Δ is sampled uniformly specified by K_2

noise regime not known to be enough for PKE