This class: “secure computation on secret data”

So far we’ve seen FHE/lattice techniques

FHE: \( \text{Encrypt}(x) \xrightarrow{\text{Eval}} \text{Dec}(y) \)

- No circuit-dependent ones
  - Narrow band of assumptions (lattice-based)
    - Conceptually: eggs in one basket
    - Practically: heavy, slow

Garbled Circuits

![Garbled Circuit Diagram]

General idea: progress through circuit gate by gate obtaining wire labels
Secret sharing based: GMW/BCW

\[ g(x) \quad \quad \quad g(y) \]
\[ g(z) \]
\[ g(x_1 + x_2) = z \]

Real idea: progress through circuit gate by gate, maintaining this invariant.

**Comparison & Contrast**

<table>
<thead>
<tr>
<th>FHE</th>
<th>GrC/GMW/BCW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Succinct comm.</td>
<td>Comm. per gate</td>
</tr>
<tr>
<td>Narrow band of assumptions (matrices)</td>
<td></td>
</tr>
</tbody>
</table>
- Conceptually, eggs in one basket
- Practically: powerful crypto runs slower |
| Generically instantiateable (OT, OWF) |
- Fast in practice with lower grade crypto (e.g., AES, elliptic curve) |
HSS: secret-sharing analogue of FHE

Bendlin 86: distributed
Boyle, Gillon
12-13: non-interactive evaluation

Can we realize this with a broader class of assumptions than FHE?
- with weaker, faster primitives
- while also maintaining meaningful efficiency
It's been a few years:

[Israël's Worlds of HSS (a la Impagliazzo)]

---

**Extremes:**

- Trivial
- Algorithmic: $P = NP$

**High-end:**

- "Centria": LWE+
- Circuits, from spooky encryption

---

Interesting stuff is what lies in between.
Cryptomania:

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

DDH construction is Eynan Telikepia

Constant degree multivariate polynomials high noise

Lapland Expand

Pseudorandom Correlation Generators

OT, OLE, etc. from LPN parameters not known to give PKE. Exciting MPC work

Minicrypt: One-way functions

Point functions ++ (intervals, decision free)

Here the results exposition framed letter as the dual notion of

Function Secret Sharing
Function Secret Sharing (FSS)

Recall
HSS: \[ x_0 \xrightarrow{\text{Eval}(f)} y_0 \]
\[ x_1 \xrightarrow{\text{Eval}(f)} y_1 \]
\[ \text{Dec} \rightarrow y = f(x_0) \]

FSS:
\[ f_0 \xrightarrow{\text{Eval}(f_0)} y_0 \]
\[ f_1 \xrightarrow{\text{Eval}(f_1)} y_1 \]
\[ \text{Dec} \rightarrow y = f(x) \]

Parameters: \( p \), \( \mathbb{N} \), parties, function class \( F \), secret shares \( i \)
Gen: \( i^n, f \rightarrow k_1, k_2, \ldots, k_p \) (p-parties)
Eval: \( i, k_i, x \rightarrow y_i \)
Dec: \( y_1, y_2, \ldots, y_d \rightarrow y \)

**Correctness:** \( \forall \ y \in \mathcal{F}, \ x \in \text{Domain}(y) \)

\[
\forall \ k_1, -k_0 \leftarrow \text{Gen}\ (\mathcal{K}, \{f\}),
\]

\[
\text{Dec}(\text{Eval}(\mathcal{C}, k_1, x), \ldots, \text{Eval}(\mathcal{C}, k_0, x)) = f(x)
\]

**Security:** \( m \)-party, \( t \)-secure FSS: for \( F \)

\[
\forall s \in \mathcal{M}, \ s.t. \ 1 \leq t, \ \exists \ \text{PPT Sim}\ s.t. \ \forall f \in \mathcal{F}, \ \text{the following are indist}: \]

Real (1^k): \( k_1, -k_0 \leftarrow \text{Gen}\ (\mathcal{K}, \{f\}) \)

output (\( k_i \)) \( \forall s \)

Ideal (1^k): output Sim (1^k, s) 

\[
\text{Real}_A \approx_c \text{Ideal}_A
\]

can include leakage
Let's rule out

Unwanted construction:

\[ \text{Gen}(f): \text{interpret } f \text{ as bit string } \in \{0, 1\}^\star \]
\[ \text{Sample } f_0, f_1 \in \{0, 1\}^\star \]
\[ f_0 \oplus f_1 = f \]
\[ \text{output } f_0, f_1 \]

\[ \text{Eval}(f_i, x): \text{output } f_i, x \land y \]

\[ \text{Dec}(g_i): \text{reconstruct } f = f_0 \oplus f_1 \]
\[ \text{output } f(x) \]

"Function privacy" from Dec

\[ \text{Gen}(f): \tilde{z}, (x_{i_0}, x_{i_1})_{i \in \{0, 1\}} \leftarrow \text{Gen}(f) \]
\[ \text{output } f_0, f_1 = (x_{i_0}, x_{i_1})_{i \in \{0, 1\}} \]

\[ \text{Eval}(f_i, f_j, x): \text{if } i = 0 \text{ output } \tilde{z} \]
\[ \text{else output } (x_{j_0}, x_{j_1})_{j \in \{0, 1\}} \]
Dec(y_0...y_k): output Ev(C,E,x)

Note: works for one evaluation for
many ends need to re-randomize C,x

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

Therefore: linear decoding procedure: Dec: E:

- succinctness: share output size comparable
- Compressibility

Properties become clear with an application

Specifically constructing FSS for class of
point functions, aka Distributed Point Function
Point function:

\[ f_{ab}(x) = \begin{cases} b & \text{if } x = a \\ 0 & \text{everywhere else} \end{cases} \]

FSS for point functions, elegant soln. to PIR (aka DPR)

Private Information Retrieval

Public database \( D \)

Client \( C \) wishes to read entry at location \( a \) without revealing \( a \) 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

\[ S_1, D \]
\[ S_2, D \rightarrow \text{size } N \]

\[ y' = \gamma y, \]

Solution with FSS for point functions

\[ \text{Define point function } f_{x,1} \]
\[ i.e., f_{x,1}(x) = \begin{cases} 1 & \text{if } x = a \\ 0 & \text{otherwise} \end{cases} \]

Note: domain of \( f \) is size of database \( N \)

2. \( K_1, K_2 \leftarrow \text{Gen}(f_{x,1}) \)
3. Send \( K_1 \) to \( S_1 \), \( K_2 \) to \( S_2 \)
\[ \delta_i = y_i = \sum_{j \in \mathbb{N}} x_j \cdot \text{Eval}(i, k_i, j) \]

- Send \( y_i \) to \( C \) just \( m \) bits
  (same size as actual output)

\[ C : \text{output } x_c = y_0 + y_1 \]

Correctness:

\[ x_c = y_0 + y_1 = \left( \sum_j x_j \cdot \text{Eval}(j, k_0, j) \right) + \left( \sum_j x_j \cdot \text{Eval}(j, k_1, j) \right) = \sum_{j \in \mathbb{N}} x_j \cdot \text{Eval}(j, k_0, j) + \text{Eval}(j, k_1, j) \]

\[ \text{for } c(j) \text{ correctness of FSS} \]
\[ \sum_{j \in \mathcal{N}} x_j \cdot f_{a_i}(i) \]

\[ \{ 1 \text{ if } i = \alpha \\ 0 \text{ otherwise} \] 

Only nonzero value in the sum is \( j = \alpha \) 

\[ = x_{\alpha} \cdot 1 \]

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

**Technical:** view of each server is simulatable, simply run \( \text{Sim}(i^*) \) of FSS scheme

View server \( i = \text{Sim}(i^*, i) \)
Efficacy:

Computation for C: 1 FSS Cen

Communication complexity

- C transmits $k_1$ to $s_i$
- $s_i$ sends $y_1$ to $C$

Key size matters!

Ruling out another trivial instantiation:

additively share the entire truth table of $f$

Let $K_1 = (k_{11}, k_{12}, \ldots, k_{1u})$

$K_2 = (k_{21}, k_{22}, \ldots, k_{2u})$

such that $K_{1i} \otimes K_{2i} = f(i)$
client work and comm. linear in N

is not interesting

interesting from this line of work:

Thus, assuming OWFs, there is a two-key FSS scheme for the family of point functions with key size $O(x \cdot \log N)$

High level idea:

$K_1, K_2$ will define CCM style binary tree with $N$ leaves where each leaf is an ‘evaluation point’

assuming a length doubling PRC:

$K_1 \quad x=01 \quad K_2$

\[ \phi \circ \neq 0 \]
\[ \phi \circ = 0 \]
In CGM: 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 PLEN additive secret sharing

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

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

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

abuse + notation

Two cases:

1) $s = 0, s_1 = s_2$

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

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

$G([s]) = (G(s_1), G(s_2)) = (\text{pseudo})\text{random}$

$= [0]$
Takeaway: $G(\ell s) \text{ expands small } \[s\text{]} \rightarrow \[0\]$  
  \(\text{small (rand)} \rightarrow [\text{rand}]\)

Idea 2: Conditional correction

Let \([s] = (s_1, s_2), \ s_1 \oplus s_2 = s \in \{0,1\}^k\)  
\([t] = (t_1, t_2), \ t_1 \oplus t_2 = t \in \{0,1\}^k\)  
“Control bit”  
\(c \in \{0,1\}^k\) : public correction word

Locally computable:\n\([s \oplus t \cdot c] = (s_1 \oplus t_1 \cdot c, s_2 \oplus t_2 \cdot c)\)

Sanity check: \(s_1 \oplus t_1 \cdot c \oplus s_2 \oplus t_2 \cdot c\)
  \(= (s_1 \oplus s_2) \oplus (t_1 \oplus t_2) \cdot c\)
  \(= s \oplus t \cdot c\)
Syntax: \([s], [t] \land c\)

Expand to build nodes:

- Apply weak homomorphic call correction

Expand to build nodes:

- \([s_L], [t_L]\)
- \([s_R], [t_R]\)

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

Invariant: \([s], [t] \in \text{special path}\)

Composition:

\([s], [t] \quad a = 101\)
Two types of parent-child expansion

Type 1:

Special:

Inactive:

\[ [s_l], [t_l] \]
\[ [s_r], [t_r] \]

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

\(C: \{0,1\}^* \rightarrow \{0,1\}^{2x+2}\)

\(C([s]) = (C(s_1), C(s_2)) = ([\sigma])\)

\(\sigma \in \{0,1\}^{2x+2}\), \(\sigma = C(s_1) \oplus C(s_2)\)

Parse \(\sigma_L || \sigma_R = \sigma\)

\(\sigma_L, \sigma_R \in \{0,1\}^{x+1}\)

Our goal: set correction word \(C\) such that

Inactive \([s_L][t_L] = [0][0]\)

Active \([s_R][t_R] = [\text{rand}][C(1)]\)
Parse \( C = C_L, C_R \in \{0,1\}^{2(\lambda+1)} \)

\[
[s_L, t_L] = [s_L \oplus t \cdot c_L]
\]

\[
[0',0] = [s_L \oplus c_L]
\]

\(: \quad c_L = \overline{s_L}
\]

What about \( c_R \)?

\[
[s_R, t_R] = [s_R \oplus t \cdot c_R]
\]

\( r \in \{0,1\} \)

\[
[r, 1] = [s_R \oplus c_R]
\]

\(: \quad c_R = \overline{s_R} \oplus \overline{r}(111)
\]

Note: for a leaf node, \( r = \beta \)

Correctness of evaluation

\[
\begin{array}{c}
[s][1] \\
\end{array}
\]

\[
\begin{array}{c}
[c_L][c_R]
\end{array}
\]

\[
\begin{array}{c}
[0][0] \\
[r][1]
\end{array}
\]
Type 2 (plain)

\[ C \]

\[ \{ s \} \{ t \} \]

\[ \{ o \} \{ o \} \]

\[ \{ s_i \} \{ t_i \} \]

\[ \{ s_R \} \{ t_R \} \]

\[ \{ o \} \{ o \} \]

\[ \{ o \} \{ o \} \]

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

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

Remember expansion is zero-preserving

\[ \sigma = C(\{ s \}) = C(\{ o \}) = \{ o \} \]

Also \[ \{ t \} = \{ o \} \]

\[ \{ \sigma \oplus t \cdot c \} = \{ o \oplus o \} = \{ o \} \]

Independent of correction word
\[ [s_L, t_L, s_R, t_R] = [\sigma \oplus t \cdot c] = [0, 0, 0, 0] \]

\[ \Rightarrow \text{Invariant preserved} \]

Correctness
\[ [0][0] \]
\[ [0][0] \]
\[ [0][0] \]

**Putting it together:**

**Gen:** for \( \alpha, \beta \), say \( |\alpha| = 3 \) bits
\[ [s] = (s_1, s_2) \leftarrow \{0, 1\}^{2^3}, \ [t] \leftarrow \{0, 1\}^2 \]

\[ [s'] = " \]

\[ [s''] = " \]

\[ [s''' ] = [\beta ] \]
Output $K$-tuples $K_1 = \mathbf{s}_1 \cdot c \cdot c' \cdot c''$

$K_2 = \mathbf{s}_2 \cdot c \cdot c' \cdot c''$

$E_{\text{val}}: i, K_c, x = x_1, x_2$

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

$C \in \{0, 1\}^{2^{2x}}$

$s_{L_i}, t_{L_i}, s_{R_i}, t_{R_i}$

$s_{L_i}^{j^1}, t_{L_i}^{j^1}, s_{R_i}^{j^1}, t_{R_i}^{j^1} = C \cdot (s_i^j) \cdot t_i^j \cdot C$

If $x_j = 0$, set $s_i^j t_i^j = s_L^j t_L^j$
else set \( s'_{i+1} e'_{i+1} = \sigma_{R_i}^{j+1} e'_{R_i}^{j+1} \)

\[ K_i = s_i, t_i, c, c', c'' \]

\[ H_0 = s_i, t_i, \sigma_L^{105} \sigma_R^{911}, \sigma_L^{105} \sigma_R^{911} || \sigma_R, \sigma_L || \sigma_{R^{911}} \]

Hybrid 0: \( \sigma_L \sigma_R = C(s_2) \oplus C(s_e) \)

Hybrid 1: \( \sigma_L \sigma_R \leftarrow \{0, 1\}^{2A+2} \)

\( H_0 \approx H_1 \) by PRC security \( (s_2 \text{ not in view}) \)

Hybrid 2: \( \sigma_L \sigma_R' \leftarrow \{0, 1\}^{2A+2} \) \( \text{independent of } s \)

Hybrid 3: \( \sigma_L', \sigma_R'' \leftarrow \{0, 1\}^{2A+2} \)

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:

Thus, assuming OWFs, $F_p$ key Fss for secret functions with key length $O(2^{m^2}2^{p^2}m)$.

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

Bonus: Silent OT

PCG: Pseudorandom Correlation Generator

Correlation Generator: GenCor: c, s
  e.g. Beaver triples.
Gen: $S_1, S_2$

Expand ($S_i$): $C_i$

$C_1, C_2 \sim \text{GenCor}(1^*)$

Oblivious Transfer:

\[ \text{S: } m_0, m_1 \quad \text{R: } b \]

But: OT needs public key operations

Impagliazzo-Rudich... ...

$\Rightarrow$ expensive

Beaver 96, [KGP] := few OTs

\[ \text{OWF} \quad \text{CorRob} \text{ (HF)} \]

$S$: $\Delta \in \{0,1\}^n$

$R$: $b \in \{0,1\}^m$, choice bits
OT Correlation: $q_i \otimes r_i = \Delta \cdot b_i$

How do we use this?

1. $H(r_i)$ serve as orps to
2. $H(r_i \otimes \Delta)$ encrypt it, message pair

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

R decrypts using $H(r_i \otimes b_i \Delta) \sim H(q_i)$

Task: compress this correlation into
short seeds $s_1, s_2$

Then: Assuming LPN, there is a
2-key PCR for OT correlation

For some integers $m, n, t$

\[ \Delta \alpha_1, \alpha_2 \ldots \alpha_t \in [m] \]

Define a point function $f(x) = \begin{cases} \Delta & \text{if } x \in E \\ 0 & \text{otherwise} \end{cases}$
FSS for multi-point keys. (lin. comb. of p.f.)
produce keys $K_1, K_2$

So on $S_1 : K_1, \alpha$

$S_2 : K_2, \Delta$

Define $L_i = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix}$

$L_i = 1$ if $i \in \alpha$  $\Rightarrow$ fully specified by $S_1$

$0$ otherwise

Define $q_i = \text{Eval}(K_1, i)$

$r_i = \text{Eval}(i, K_2, i)$

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

Looks like OR correlation, but we've not seen yet.

This is where CPN comes in.
Dual LPN assumption:

\[
\begin{align*}
\hat{v} & \leftarrow m \rightarrow \begin{bmatrix} e \end{bmatrix} \sim c \left[ \begin{bmatrix} u \end{bmatrix} \right]_n
\end{align*}
\]

binary vector of low HW
sampled according to some dist.

\(H\) is public, so matrix \(m\) is a linear operation on the shares

\[
\text{Expand } (s_i = \tilde{r}_i, \tilde{\alpha}) \quad \rightarrow \quad \tilde{E} (s_i = K_i, \Delta)
\]

\[
\begin{align*}
H \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix} &= \begin{bmatrix} q_1' \\ \vdots \\ q_m' \end{bmatrix} \\
H \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix} &= \begin{bmatrix} r_1' \\ \vdots \\ r_m' \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
q_i' &= \langle H_i, q_i \rangle \\
r_i' &= \langle H_i, r_i \rangle = \langle H_i, 0 \Delta (H_i \cdot b_i) \rangle = q_i \oplus \Delta (H_i \cdot b_i)
\end{align*}
\]
\[ v_i \oplus \Delta \cdot \hat{v}_i \]

By dual-LPN, \( \tilde{v} \) is pseudorandom & specified by \( K_1 \)

\( \Delta \) is sampled uniformly specified by \( K_2 \)

Noise regime not known to be enough for PKE