Beaver Meets FSS:
Secure Computation with Preprocessing via Function Secret Sharing

Elette Boyle
IDC Herzliya

Niv Gilboa
Ben Gurion University

Yuval Ishai
Technion
Goal: Efficient 2PC with Preprocessing

• Possibly mixed domains (big)

• Useful nonlinear gates
  • Equality, Comparison, Bit Decomp, …
Secure Computation with Preprocessing

[Beaver ’91]

Correlated randomness

Preprocessing

Online phase

\( f(x, y) \)

\( x \)

\( y \)

- Cheap
- Low communication
Semi-Orthogonal Questions

• How to **use** correlations (& which are useful)?
  • Beaver triples, circuit-dependent Beaver [Bea91]
  • One-time truth tables (TinyTables) [IKMOP13, DNNR17]
  • Sublinear IT online comm for layered circuits [Cou19]
  • …

This Talk

• How to **generate** correlations?

  New & cool! “Silent Preprocessing”
  [BCGIKS19, BCGIKRS19]
Our Results (High Level)

• General Framework: MPC w Preprocessing via Function Secret Sharing

- Theoretical: Unifying approach
- Practical: Promising low-online-comm (equality, comparison, bit decomp, …)

• Necessity of FSS?
  - “Shared equality” with optimal online communication ⇒ OWF
  - Barrier to such implication for almost optimal
Today

• Function Secret Sharing (FSS)
• General Transformation
• Examples (Old & New)
Additive Secret Sharing

Elements in Abelian group $\mathbb{G}$

$s_0 + s_1 = s$ (in $\mathbb{G}$)

- **Secrecy**: $s_b$ hides $s$
- **Reconstruction**: $s_0 + s_1 = s$ (in $\mathbb{G}$)
Function Secret Sharing (FSS) for $\mathcal{F}$

- **Security**: $f_b$ hides $f$ (within $\mathcal{F}$)
- **Correctness**: For every input $x$, $\text{Eval}_x(f_0) + \text{Eval}_x(f_1) = f(x)$
- **Size**: $|f_b| \sim |f|$
Our Preprocessing Framework: 2 Flavors

- Circuit **Dependent**
- Circuit **Independent**
General Framework

- Step 1: Choose circuit

Each gate $g$:

$F_g : G^\text{in}_g \rightarrow G^\text{out}_g$

Each wire $w$:

Group $G_w$

Note: Can also mix & match with other frameworks
**General Framework**

- **Step 1:** Choose circuit
- **Step 2:** Random mask per wire
  \[ r_w \leftarrow \mathcal{G}_w \]
- **Online goal:** Parties progressively learn masked values
  \[ \hat{x}_w := x_w + r_w \]
- **Inputs:** \( r_w \in \text{party's CR}. \) He sends \( \hat{x}_w \).
- **Gates:** Need translation
  \[ \hat{x}_{\text{out}} = F_g(\hat{x}_{\text{in}} - r_{\text{in}}) + r_{\text{out}} \]
Offset-Function Family

\[ \hat{x}_{out} = F_g(\hat{x}_{in} - r_{in}) + r_{out} =: F_g^{r_{in}, r_{out}}(\hat{x}_{in}) \]

Offset \((F_g) := \left\{ F_g^{r_{in}, r_{out}} : r_{in} \in G^\text{in}_g, \quad r_{out} \in G^\text{out}_g \right\} \)
Leveraging FSS for Offset($F_g$)

Public: $\hat{x}_{in}$

Goal: $\hat{x}_{out}$

\[
\begin{align*}
F_g & = F_{[r_{in}, r_{out}]}(\hat{x}_{in}) \\
& = F_g(\hat{x}_{in} - r_{in}) + r_{out} \\
& = \hat{x}_{out}
\end{align*}
\]
Combining Gates

FSS shares for each gate + Input $r_w$’s

Inputs: Send masked $\hat{x}_w$

Gate: Exchange FSS-Eval’ed shares of $\hat{x}_{out}$
Circuit-Dependent Preprocessing

- **Offline:** Size ~ \{all FSS shares\}

- **Online:**
  - Rounds = 1 per FSS gate
  - Comp = FSS Evals
  - Comm = \( \sum_g |G_g^{out}| \)
Circuit-Independent Preprocessing

• Don’t know how to “match up” $r_{in}$ & $r_{out}$'s

\[ F_1[r_{1in}, r_{1out}] \]
\[ F_2[r_{2in}, r_{2out}] \]
**Circuit-Independent Preprocessing**

- Independently sample $r_{in}$'s

  \[ F_{g[r_{in}, 0]}(\hat{x}) = F_g(\hat{x} - r_{in}) + 0 \]

- Also give additive shares of $r_{in}$'s

\[ \langle r_{2_{in}} \rangle \Rightarrow \text{Shares of unmasked } x \]
Circuit-independent Preprocessing

- **Offline:**
  
  \[
  \text{Size} \sim \{\text{all FSS shares}\} + \sum_g |G_{g}^{in}| 
  \]

- **Online:**
  
  - Rounds = 1 per FSS gate
  - Comp = FSS Evals
  - Comm = \( \sum_g |G_{g}^{in}| \)

Different (higher) b/c fan-out
So... About that FSS.
Examples: Information-Theoretic FSS

- **Any** function class \( \{ f : \{0,1\}^n \rightarrow \mathbb{G} \} \)
  - Secret share the truth table

- Low-degree **polynomials** \( \{ \sum_i \alpha_i x^i \} \)
  - Secret share the coefficients \( \alpha_i \)

- Function class \( \{ \sum_i \alpha_i f_i(x) \} \) for **public** \( f_i \)
  - Secret share the coefficients \( \alpha_i \)
Corollaries

- **Any** function class \( \{ f : \{0,1\}^n \to \mathbb{G} \} \)
  - Secret share the truth table

- Low-degree **polynomials** \( \{ \sum_i \alpha_i x^i \} \)
  - Secret share the coefficients \( \alpha_i \)

- Function class \( \{ \sum_i \alpha_i f_i(x) \} \) for **public** \( f_i \)
  - Secret share the coefficients \( \alpha_i \)

One-time truth tables [IKMOP13]
TinyTables [DNNR17]
(TT for local functions) [Cou19]

Beaver triples [Bea91]
Circuit-dependent Beaver [DNNR17]

\[(x_1 - r_1)(x_2 - r_2) = x_1x_2 - r_1x_2 - x_1r_2 + r_1r_2\]

Degree-\(d\) gates
Bilinear maps, …
Lightweight FSS Constructions from OWF

- **Point Functions** $f_{\alpha, \beta} : \{0,1\}^n \to \mathbb{G}$
  - Key size $\sim \lambda n + \log |\mathbb{G}|$ bits
  - Gen/Eval $\sim n$ PRG evals

- **“Special” Intervals**
  - Cost $\leq$ Point Function $\times 2$

- **General Intervals**
  - Cost $\leq$ Point Function $\times 4$

*General input groups too*
Zero Test / Equality Match

- \( F_{zt}(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{else} \end{cases} \)

- \( F_{zt}^{[r_{in}, r_{out}]}(\hat{x}) = r_{out} + \begin{cases} 1 & \hat{x} = r_{in} \\ 0 & \text{else} \end{cases} \)

• Equality Match = ZeroTest(x-y)
Sign / Comparison

- $F_{\text{sign}}(x) = \begin{cases} 
1 & x \geq 0 \\
0 & \text{else} 
\end{cases}$

- $F_{\text{sign}}^*[r,0](\hat{x}) = \begin{cases} 
1 & \hat{x} - r \geq 0 \\
0 & \text{else} 
\end{cases}$

- Comparison $= \text{Sign}(x-y)$

sum: 2 Special Intervals
($\sim 4 \times \text{Point Fn}$)
Splines

\[ F(x) = (x \in I_1) \cdot 0 + (x \in I_2) \cdot x + (x \in I_3) \cdot \sum \alpha_i x^i \]

\[ F_{\text{piece}}(x) = (x \in I) \cdot p(x) \]

\[ F^{[r,0]}_{\text{piece}}(\hat{x}) = (\hat{x} \in I_r) \cdot p(\hat{x} - r) \]

- Example: ReLU. Needs single interval of payload = 2 coeffs

Polynomial, secret coefficients \( \beta_i \)

Interval, where “yes” payload is vector of \( \beta_i \)’s
Bit Decomposition: \( \mathbb{Z}_{2^k} \)

- Goal: FSS BitDecomp( \( \hat{x} + r \) )

\[ \hat{x}_i \oplus r_i \oplus \text{Carry}_i^r(\hat{x}) \]

Linear

\[ \equiv [\hat{x}]_{i-i} \geq 2^i - [r]_{i-1} \]

Special Interval!!

- All together: \( k \) Special intervals. Total size/complexity \( \sim \lambda k^2 \)
Bit Decomposition: General $\mathbb{Z}_N$

Challenge: Wraparound

• Case 1: General N, with small payload guarantee ($|x| \leq \frac{1}{2} \log N$)
  • (Removes issue)

• Case 2: General N (no promise)
  • Compute as 2 conditionals = (wraparound $\land$ carry) + ($\neg$ wraparound $\land$ carry)
  • Each is 2-dim interval (less cheap)
# Example Comparison

<table>
<thead>
<tr>
<th>Gate Type</th>
<th>Protocol</th>
<th>Online communication (bits per party)</th>
<th>Online rounds</th>
<th>Offline storage (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero test</td>
<td>[Cou18] ABY [DSZ15] This work</td>
<td>$m + o(m)$ [O(\lambda m)] [m]</td>
<td>$\geq 3$ [2]</td>
<td>$2m + o(m)$ [O(\lambda m)] [\approx \lambda m]</td>
</tr>
<tr>
<td>Zero test example $m = 64$</td>
<td>[Cou18] This work</td>
<td>77 [64]</td>
<td>$3$ [1]</td>
<td>152 [8322]</td>
</tr>
<tr>
<td>Integer comparison</td>
<td>[Cou18] SC1 ABY [DSZ15] This work</td>
<td>[O(\lambda m)] [m]</td>
<td>$O(\log \log m)$ [2]</td>
<td>$3m + o(m)$ [O(\lambda m)] [\approx \lambda m]</td>
</tr>
<tr>
<td>Comparison example $m = 64$</td>
<td>[Cou18] This work</td>
<td>1120 [64]</td>
<td>$12$ [1]</td>
<td>$\approx 300$ [8450]</td>
</tr>
<tr>
<td>Bit decomposition</td>
<td>ABY [DSZ15] This work</td>
<td>[O(\lambda m)] [m]</td>
<td>$2$ [1]</td>
<td>$O(\lambda m)$ [\approx \lambda m^2/2]</td>
</tr>
<tr>
<td>Spline over $\mathbb{Z}_2^m$ $k + 1$ deg.-$d$ polynomials</td>
<td>ABY [DSZ15] This work</td>
<td>[O(m(\lambda k + d))] [m]</td>
<td>$2$ [1]</td>
<td>$O(m(\lambda k + d))$ [\approx 2km(\lambda + d)]</td>
</tr>
</tbody>
</table>
Generating the Correlations

• **2-Party**: Secure 2PC of FSS for Point Functions
  • Via Generic 2PC (~2n AES evals)
  • Via Doerner-shelat [Ds17] for reasonable domains (BB in PRG)

• **3-Party**: One party simply generates & goes offline
Summary of Results

• General Framework:  MPC w Preprocessing via Function Secret Sharing

• Theoretical:  Unifying approach

• Practical:  Promising low-online-comm (equality, comparison, bit decomp,...)

• Necessity of FSS?
  • “Shared equality” with optimal online communication ⇒ OWF
  • Barrier to such implication for almost optimal
Open Directions

- **Supporting further gates**
  - New efficient FSS schemes (e.g., CNF/DNF from OWF?)
  - New tricks for using FSS to attain more gates

- **Efficiency**
  - Eg: improved Bit Decomposition \((\text{better than quadratic: } \lambda \cdot k^2)\)
  - Concrete optimizations & implementation

- **Improved generation of correlations**
  - Targeted 2PC of FSS Share function
  - Silent generation??