Scalable Multiparty Garbling

Gabrielle Beck  Aarushi Goel  Aditya Hegde  Abhishek Jain  Zhengzhong Jin  Gabriel Kaptchuk
Secure Multiparty Computation (MPC)

\[ y = f(x_1, x_2, x_3, x_4) \]
Secure Multiparty Computation (MPC)

\[ y = f(x_1, x_2, x_3, x_4) \]

Adversary learns nothing beyond input of corrupt parties and function output
Applications Of MPC

- Machine learning on distributed datasets
- Data as a Service (DaaS)
- Securing digital assets and key management
Applications Of MPC

Machine learning on distributed datasets

Data as a Service (DaaS)

Securing digital assets and key management

Large computations over the internet
MPC Over The Internet

Over the internet, network latency limits protocol runtime [WRK17b]
MPC Over The Internet

Over the internet, network latency limits protocol runtime [WRK17b]

Constant Round MPC: Parties interact constant number of times

- Two approaches: FHE-based and Multiparty Garbling
Problem: Getting Best Of Both Worlds

$n \rightarrow$ Number of parties
$|C| \rightarrow$ Size of circuit

Dishonest majority, multiparty garbling protocols with total communication cost of $O(n^2|C|)$

[HSS17, WRK17b, BCOOSS21]
Problem: Getting Best Of Both Worlds

\[ n \to \text{Number of parties} \]
\[ |C| \to \text{Size of circuit} \]

### Dishonest majority, multiparty garbling protocols with total communication cost of \( O(n^2 |C|) \)
- [HSS17, WRK17b, BCOOSS21]

### Honest majority, non-constant round MPC with total communication cost of \( O(|C|) \)
- [BGJK21, GSY21, GPS21, GPS22]
Problem: Getting Best Of Both Worlds

Dishonest majority, multiparty garbling protocols with total communication cost of $O(n^2 |C|)$

[HSS17, WRK17b, BCOOSS21]

Honest majority, non-constant round MPC with total communication cost of $O(|C|)$

[BGJK21, GSY21, GPS21, GPS22]

$n \rightarrow$ Number of parties
$|C| \rightarrow$ Size of circuit

Is there a multiparty garbling protocol with $O(|C|)$ communication in the honest majority setting?
MPC Protocols With $O(|C|)$ Communication

- Per party communication decreases as the number of parties increases.

- Scales to hundreds of parties and can be used with large volunteer networks like Tor and Bitcoin.

- Honest majority is a more plausible assumption.
Our Contributions

- Semi-honest and maliciously secure
- \( t < n \left( \frac{1}{2} - \varepsilon \right) \) where \( 0 < \varepsilon < \frac{1}{2} \)
- Based on Learning Parity with Noise (LPN) over large fields assumption
- Benchmarks and evaluation show that our protocol is practical

**Multiparty garbling protocol with** \( O(|C|) \) **communication complexity**
Outline

Template for multiparty garbling
Overview of prior works
Key techniques in our protocol
Benchmarks and evaluation
Review Of Garbled Circuits [Yao86]

Garbling

\[
\begin{align*}
&C \quad \rightarrow \quad \text{Garble} \quad \rightarrow \quad \hat{C} \\
&x \quad y \\
&z
\end{align*}
\]

\[
\begin{align*}
&X_0 \quad Y_0 \\
&X_1 \quad Y_1
\end{align*}
\]
Review Of Garbled Circuits [Yao86]

Garbling

\[ C \rightarrow \text{Garble} \rightarrow \hat{C} \]

\[ x \ y \ z \]

Evaluation

\[ \hat{C} \rightarrow \text{Eval} \rightarrow c \]

\[ X_0 \ Y_0 \ X_1 \ Y_1 \]

\[ c = C(a, b) \]
Review Of Garbled Circuits [Yao86]

Adversary having only $\hat{C}$ and one label per wire does not learn anything beyond the output $c = C(a, b)$.
Review Of Garbled Circuits [Yao86]

Garbling

Adversary having only \( \hat{C} \) and one label per wire does not learn anything beyond the output.
Template For Multiparty Garbling [BMR90]

Garbling Phase

\(a\) \(b\)
Template For Multiparty Garbling [BMR90]
Template For Multiparty Garbling [BMR90]

Garbling Phase

\[ \hat{C} X_a Y_b \]

Evaluation Phase

\[ \hat{C} X_a Y_b \]
\[ \hat{C} X_a Y_b \]
\[ \hat{C} X_a Y_b \]
\[ \hat{C} X_a Y_b \]
Template For Multiparty Garbling [BMR90]

Garbling Phase

Evaluation Phase

Any non-constant round MPC works, since garbling algorithm can be computed by constant depth circuit
Template For Multiparty Garbling [BMR90]

Garbling Phase

Evaluation Phase

Any non-constant round MPC works, since garbling algorithm can be computed by constant depth circuit

MPC ensures adversary only learns \( \hat{C} \) and one label per wire
Template For Multiparty Garbling [BMR90]

Garbling Phase

Evaluation Phase

\[ \hat{C} X_a Y_b \]

MPC ensures adversary only learns \( \hat{C} \) and one label per wire

Any non-constant round MPC works, since garbling algorithm can be computed by constant depth circuit

Inefficient due to non-black box use of encryption
Avoiding Non-Black Box Use Of Encryption

Parties locally evaluate the PRF used in encryption [DI05]

\( \hat{C} \) is linear in \( n \)
Avoiding Non-Black Box Use Of Encryption

Parties **locally** evaluate the PRF used in encryption \([\text{DI05}]\)

\[ \widehat{C} \text{ is linear in } n \]

Parties **locally** compute additive sharing of ciphertext from additive sharing of the key and message i.e., \(E([k], [m]) \rightarrow [E(k, m)] \) [BLO17]

MPC protocols work with secret shares

\([x] \rightarrow \text{secret sharing of } x\)
Avoiding Non-Black Box Use Of Encryption

Parties **locally** evaluate the PRF used in encryption [DI05]

\[ |\hat{C}| \text{ is linear in } n \]

Parties **locally** compute additive sharing of ciphertext from additive sharing of the key and message i.e., \( E([k], [m]) \rightarrow [E(k, m)] \) [BLO17]

\[ |\hat{C}| \text{ is independent of } n \]

MPC protocols work with secret shares

\[ [x] \rightarrow \text{secret sharing of } x \]
Avoiding Non-Black Box Use Of Encryption

Parties **locally** evaluate the PRF used in encryption [DI05]

\[ |\mathcal{C}| \text{ is linear in } n \]

Parties **locally** compute **additive sharing** of ciphertext from **additive sharing** of the key and message i.e., \( E([k], [m]) \rightarrow [E(k, m)] \) [BLO17]

\[ |\mathcal{C}| \text{ is independent of } n \]

MPC protocols work with **secret shares**

\[ |\mathcal{C}| \text{ is independent of } n \]

\( O(n^2 |\mathcal{C}|) \) communication due to dishonest majority MPC
Avoiding Non-Black Box Use Of Encryption

Parties locally evaluate the PRF used in encryption \[\text{[DI05]}\]

\[\widehat{C}\] is linear in \(n\)

Parties locally compute additive sharing of ciphertext from additive sharing of the key and message i.e., \(E([k], [m]) \rightarrow [E(k, m)]\) \[\text{[BLO17]}\]

\(\widehat{C}\) is independent of \(n\)

\(O(n^2|C|)\) communication due to dishonest majority MPC

Can we leverage techniques from \(O(|C|)\) communication honest majority, non-constant round MPC?
Achieving $O(|C|)$ Communication

Parties locally compute threshold sharing of ciphertext from threshold sharing of key, message, and randomness i.e., $E([k], [m]; [r]) \rightarrow [E(k, m; r)]$

Based on Learning Parity with Noise (LPN)
Achieving $O(|C|)$ Communication

Parties locally compute threshold sharing of ciphertext from threshold sharing of key, message, and randomness i.e., $E([k], [m]; [r]) \rightarrow [E(k, m; r)]$

Based on Learning Parity with Noise (LPN)

$|C|$ is independent of $n$  
Compatible with honest majority MPC
Achieving $O(|C|)$ Communication

Parties locally compute threshold sharing of ciphertext from threshold sharing of key, message, and randomness i.e., $E([k], [m]; [r]) \rightarrow [E(k, m; r)]$

Based on Learning Parity with Noise (LPN)

- $|C|$ is independent of $n$
- Compatible with honest majority MPC

Packed secret sharing [FY92] - Standard technique in $O(|C|)$ communication non-constant round MPC protocols [BGJK21, GSY21, GPS21, GPS22]

- Pack $O(n)$ secrets into a single sharing $\Rightarrow$ Reduces communication cost by a factor of $n$
Achieving $O(|C|)$ Communication

Parties locally compute threshold sharing of ciphertext from threshold sharing of key, message, and randomness i.e., $E([k], [m]; [r]) \rightarrow [E(k, m; r)]$

Based on Learning Parity with Noise (LPN)

$|C|$ is independent of $n$  
Compatible with honest majority MPC

Packed secret sharing [FY92] - Standard technique in $O(|C|)$ communication non-constant round MPC protocols [BGJK21, GSY21, GPS21, GPS22]

- Pack $O(n)$ secrets into a single sharing $\Rightarrow$ Reduces communication cost by a factor of $n$
- Utilize existing techniques for MPC over packed sharings
Achieving $O(|C|)$ Communication

Parties locally compute threshold sharing of ciphertext from threshold sharing of key, message, and randomness i.e., $E([k], [m]; [r]) \rightarrow [E(k, m; r)]$

Based on Learning Parity with Noise (LPN)

$|C|$ is independent of $n$  

Compatible with honest majority MPC

Packed secret sharing [FY92] - Standard technique in $O(|C|)$ communication non-constant round MPC protocols [BGJK21, GSY21, GPS21, GPS22]

- Pack $O(n)$ secrets into a single sharing $\implies$ Reduces communication cost by a factor of $n$
- Utilize existing techniques for MPC over packed sharings
- Efficiently computing $[r]$ requires developing new subprotocols, building on prior works [CCXY18]
Evaluation Of Semi-Honest Secure Protocol

\[ n = 512 \quad t = 127 \quad 2 \text{ threads per party} \]

| \(|C| \) | \(\equiv\) | Runtime | Per Party Communication |
|------|---------|---------|------------------------|
| 36,663 | AES-128 | 126.28 s | 66.84 MB |
| 114,107 | SHA-256 | 481.88 s | 240.44 MB |
Evaluation Of Semi-Honest Secure Protocol

\[ n = 512 \quad t = 127 \quad 2 \text{ threads per party} \]

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Total</th>
<th>Circuit Dependent</th>
<th>Runtime</th>
<th>Per Party Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \equiv \text{AES-128} )</td>
<td>126.28 s</td>
<td>15.52 s</td>
<td>66.84 MB</td>
<td>8.1 MB</td>
</tr>
<tr>
<td>(</td>
<td>C</td>
<td>= 36,663)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Total</th>
<th>Circuit Dependent</th>
<th>Runtime</th>
<th>Per Party Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \equiv \text{SHA-256} )</td>
<td>481.88 s</td>
<td>40.8 s</td>
<td>240.44 MB</td>
<td>27.07 MB</td>
</tr>
<tr>
<td>(</td>
<td>C</td>
<td>= 114,107)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Evaluation Of Maliciously Secure Protocol

$n = 512$ \hspace{1cm} $t = 127$ \hspace{1cm} 2 threads per party

<table>
<thead>
<tr>
<th>Circuit Type</th>
<th>Computation Time</th>
<th>Per Party Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total</strong></td>
<td>$\approx 220 \text{ s}$</td>
<td>$\approx 334 \text{ MB}$</td>
</tr>
<tr>
<td><strong>Circuit Dependent</strong></td>
<td>$\approx 18 \text{ s}$</td>
<td>$\approx 42 \text{ MB}$</td>
</tr>
</tbody>
</table>

\[|C| = 36,663\]

\[\equiv \text{AES-128}\]

\[\equiv \text{SHA-256}\]

<table>
<thead>
<tr>
<th>Circuit Type</th>
<th>Computation Time</th>
<th>Per Party Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total</strong></td>
<td>$\approx 811 \text{ s}$</td>
<td>$\approx 1230 \text{ MB}$</td>
</tr>
<tr>
<td><strong>Circuit Dependent</strong></td>
<td>$\approx 67 \text{ s}$</td>
<td>$\approx 155 \text{ MB}$</td>
</tr>
</tbody>
</table>

\[|C| = 114,107\]
Thank You

ia.cr/2023/099

github.com/adishegde/scalable_garbling
Comparison of per party communication when garbling AES-128