Faster Malicious 2-party Secure Computation with Online/Offline Dual Execution
2 Party Computation

\[ f(x, y) \]

Real Protocol

Ideal Functionality

\[ f(x, y) \]
2 Party Computation

- Secure against malicious adversaries
- Def (simplified):

\[
\forall \exists \exists \exists : \mathcal{S} (f, y, f(x, y)) \approx \mathcal{S} (f, y, f(x, y))
\]
Applications

2-party Secure Computation

\[ f(x, y) = f(x, y) \]

Applications

- Private database querying
  - Database \( x \),
  - Query \( y \)

- Joint machine learning
  - Datasets \( x, y \),
  - Model = \( f(x, y) \)

- Secure auctions
  - Bids \( x, y \)
  - Winning bid = \( f(x, y) \)
Yao’s Protocol

\[ f(x, y) \]

\[ \text{def} \]

\[ \text{garbled } f(x, y) \]

- Security properties:
  - Privacy – Alice learn no more than \( f(x, y) \)
  - Authenticity – Alice cannot guess any output encoding other than \( [f(x, y)] \)
Yao’s Protocol

Problems with malicious Adversaries

- The circuit may not be correctly constructed
  - E.g. $g(x) := x$
- May violate privacy and correctness
- Not always detectable
Dual Execution [MohasselFranklin06]

- First Yao secure against Alice.
- Second Yao secure against Bob
Dual Execution \[ \text{[MohasselFranklin06]} \]

- Define common encoding:

\[
[Z]_{AB} \overset{\text{def}}{=} [Z]_A \oplus [Z]_B
\]
Dual Execution \[\text{[MohasselFranklin06]}\]

- Define common encoding:
  \[
  [z]_{AB} \overset{\text{def}}{=} [z]_A \oplus [z]_B
  \]

- Leaks a single bit!
  \[
  g(x) \neq f(x, y)
  \]

- Guaranteed Correctness
Improved Dual Execution [KolesnikovMohasselRivaRosulek15]

- Send $s$ circuits
- Check some for correctness
Improved Dual Execution [KolesnikovMohasselRivaRosulek15]

• Send $s$ circuits
• Check some for correctness
• Use Private Set Intersection (PSI) to reconcile
Improved Dual Execution [KolesnikovMohasselRivaRosulek15]

- Send $s$ circuits
- Check some for correctness
- Use Private Set Intersection (PSI) to reconcile

$[f(x,y)]_{AB}$ or $\emptyset$
Improved Dual Execution [KolesnikovMohasselRivaRosulek15]

- Send $s$ circuits
- Check some for correctness
- Use Private Set Intersection (PSI) to reconcile

Leaks a bit iff all eval. Circuit are bad:

$$\forall i : g_i(x) \neq f(x, y)$$

$$\Pr[\text{leak a bit}] = 2^{-s}$$
• Want to perform $N$ executions of $f$
  • Construct enough circuits for all $N$ executions
Online – Offline [LindellRiva14,NeilsenOrlandi08,Rosulek16]

- Want to perform $N$ executions of $f$
  - Construct enough circuits for all $N$ executions
  - Check some for correctness
Online – Offline [LindellRiva14, NeilsenOrlandi08, Rosulek16]

- Want to perform $N$ executions of $f$
  - Construct enough circuits for all $N$ executions
  - Check some for correctness

- Randomly map the rest into bins
  - Bin size of $\frac{s}{\log N}$ instead of $s$
  - E.g. 10 $\times$ improvement
Online – Offline [LindellRiva14, NeilsenOrlandi08, Rosulek16]

- Want to perform $N$ executions of $f$
  - Construct enough circuits for all $N$ executions
  - Check some for correctness

- Randomly map the rest into bins
  - Bin size of $s / \log N$ instead of $s$
  - E.g. 10 $\times$ improvement
Online – Offline

[LindellRiva14, NeilsenOrlandi08, Rosulek16]

- Want to perform $N$ executions of $f$
  - Construct enough circuits for all $N$ executions
  - Check some for correctness

- Randomly map the rest into bins
  - Bin size of $s / \log N$ instead of $s$
  - E.g. $10 \times$ improvement
Online – Offline [LindellRiva14, NeilsenOrlandi08, Rosulek16]

- Use one bin per evaluation

\[ f(x, y) \]
Challenge #1: Input Consistency

How to ensure Bob used the same $y$ in all circuits?
Challenge #1: Input Consistency [RRosulek16]

How to ensure Bob used the same $y$ in all circuits?

- Circuit generated by Alice
  - Bob receives input via OT [easy]

![Diagram showing PSI protocol]
Challenge #1: Input Consistency

How to ensure Bob used the same $y$ in all circuits?

• Circuit generated by Alice
  • Bob receives input via OT [easy]

• Circuit generated by Bob [hard]
Challenge #1: Input Consistency \[RRosulek\]16

How to ensure Bob used the same \( y \) in all circuits?

- Circuit generated by Alice
  - Bob receives input via OT \[easy\]

- Circuit generated by Bob \[hard\]
  - In the offline, Bob tells Alice the relationship \( \bowtie \) between the two arrows
  - Alice checks \( \bowtie \) in the cut and choose
Challenge #1: Input Consistency \[ \text{[RRosulek16]} \]

How to ensure Bob used the same \( y \) in all circuits?

- Circuit generated by Alice
  - Bob receives input via OT \([\text{easy}]\)

- Circuit generated by Bob \([\text{hard}]\)
  - In the offline, Bob tells Alice the relationship \( \mathcal{G} \) between the two arrows
  - Alice check \( \mathcal{G} \) in the cut and choose

- Consistent with the relationship \( \Rightarrow \) at least of one of Bob’s circuits uses \( y \)
  - Requires \textit{no crypto operations}
Challenge #2: Private Set Intersection (PSI) [Rosulek16]

- Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  - Fastest PSI protocol
Challenge #2: Private Set Intersection (PSI)  

- Build PSI from Private Equality Test  
  - Fastest PSI protocol
- Issues: Not malicious secure in general  
  - Can not be simulated
Challenge #2: Private Set Intersection (PSI)  

- Build PSI from Private Equality Test  \([\text{PinkasSchneiderZohner14}]\)  
  - Fastest PSI protocol

- Issues: Not malicious secure in general  
  - Can not be simulated  
  - Ex: singleton set

\[ X = \{a, b, c\} \]
Challenge #2: Private Set Intersection (PSI) [Rosulek16]

• Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  • Fastest PSI protocol

• Issues: Not malicious secure in general
  • Can not be simulated
  • Ex: singleton set

\[ X = \{a, b, c\} \]
Challenge #2: Private Set Intersection (PSI)

- Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  - Fastest PSI protocol
- Issues: Not malicious secure in general
  - Can not be simulated
  - Ex: singleton set
Challenge #2: Private Set Intersection (PSI) [Rosulek16]

- **Ideal:** Bob only knows one valid PSI input
  \[ f(x, y) \]

- **Simulator doesn’t need to extract Bob input!**
  - Just test if it contains \( f(x, y) \)

- **Build PSI from Private Equality Test** [PinkasSchneiderZohner14]
  - Fastest PSI protocol

- **Issues:** Not malicious secure in general
  - Can not be simulated
  - Ex: singleton set

- **Issues:** Not malicious secure in general
  - Can not be simulated
  - Ex: singleton set
### Performance

<table>
<thead>
<tr>
<th>Function</th>
<th>[RRosulek16]</th>
<th>[LindellRiva15]</th>
<th>[DamgårdZakarias15]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline</td>
<td>Online</td>
<td>Offline</td>
<td>Online</td>
</tr>
<tr>
<td>AES</td>
<td>5.1 ms</td>
<td>74 ms</td>
<td>high?</td>
</tr>
<tr>
<td>SHA-256</td>
<td>48.0 ms</td>
<td>206 ms</td>
<td>6 ms</td>
</tr>
</tbody>
</table>

- Amortized cost for $N = 1,024$ evaluations
  - Amazon c4.8xLarge = 36 core, 64GB RAM
  - Statistical security $\kappa = 40$

- Maximum **throughput**: $0.26 \text{ ms} / \text{AES block}$ (3800+ Hz)
  - [DamgårdZakarias15] report 0.4 ms
Total Protocol Times for AES

- **[PSSW]**: 18 min
- **[KSS]**: 3.4 sec
- **[FN]**: 0.8 sec
- **[LR15]**: 81 ms
- **[RR16]**: 6 ms
Total Protocol Times for AES
Total Protocol Times for AES

- 2008: 18 min
- 2009: 3.4 sec
- 2010: 0.8 sec
- 2011: 81 ms
- 2012: 6 ms

- [PSSW] 18 min
- [KSS] 3.4 sec
- [FN] 0.8 sec
- [LR15] 81 ms
- [RR16] 6 ms

64-node cluster
Consumer GPU
AWS c4.8xLarge
Summary

• Online-offline dual execution
  • Faster 2PC with malicious security to date: $1.3ms$ AES
  • Some security advantages over “classic” cut-and-choose

• Future Work:
  • Hybrid protocols: combine [Rosulek16] with [DamgårdZakarias15]
    • fast offline
    • function independent offline
  • Transfer advances from online-offline to single execution setting
The End

Thanks

Faster Malicious 2-party Secure Computation with Online/Offline Dual Execution

github.com/osu-crypto/batchDualEx