Multi-Party Computation of Polynomials & Branching Programs without Simultaneous Interaction

Hoeteck Wee (ENS)

Dov Gordon (ACS)
Tal Malkin (Columbia)
Mike Rosulek (OSU)
multi-party computation

\[ f(x_1, x_2, \ldots, x_n) \]

- electronic voting
- secure auctions
multi-party computation

\[ f(x_1, x_2, \ldots, x_n) \]

- electronic voting
- secure auctions
multi-party computation on the web

\[ f(x_1; x_2; \ldots; x_n) \]

limited interaction e.g. web users, program committees

[Ibrahim Kiayias Yung Zhou 09, Halevi Lindell Pinkas 11]
multi-party computation on the web

\[ f(x_1, x_2, \ldots, x_n) \]

“one-pass” secure computation. [Halevi Lindell Pinkas 11]

- each party interacts once with server in fixed order
- server announces result
multi-party computation on the web

\[ f(x_1; x_2; \ldots; x_n) \]

“one-pass” secure computation. [Halevi Lindell Pinkas 11]

- each party interacts once with server in fixed order
- server announces result
- server may be corrupt and colluding with parties
  - new technical challenge beyond standard MPC
security: inherent leakage

$S$ colludes with last $k$ parties:

$\begin{align*}
&\mathbf{P}_1 \xrightarrow{x_1} S \xrightarrow{z_{n-k+1}} \mathbf{P}_2 \xrightarrow{x_2} \mathbf{P}_3 \xrightarrow{x_3} \mathbf{P}_4 \xrightarrow{x_4} f(x_1; \ldots; x_{n-k}; \star) \\
&\text{Repeatedly:} \quad \mathbf{P}_1 \xrightarrow{x_1} S \xrightarrow{z_{n-k+1}} \mathbf{P}_2 \xrightarrow{x_2} \mathbf{P}_3 \xrightarrow{x_3} \mathbf{P}_4 \xrightarrow{x_4} f(x_1; \ldots; x_{n-k}; z_{n-k+1}; \star) \\
&\text{Standard:} \quad \text{single evaluation of } f \text{ here: multiple evaluations of } f.
\end{align*}$
security: inherent leakage

\[ f(x_1, x_2, x_3, z_4), \]
\[ f(x_1, x_2, x_3, z'_4), \]
\[ f(x_1, x_2, x_3, z''_4), \]
\[ \ldots \]

\( S \) colludes with last \( k \) parties:

Repeatedly:

- run protocol on choice of \( z_{n-k+1}, \ldots, z_n \)
- learn \( f(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n) \)
security: inherent leakage

\[ f(x_1, x_2, x_3, z_4), \]
\[ f(x_1, x_2, x_3, z'_4), \]
\[ f(x_1, x_2, x_3, z''_4), \]
\[ \cdots \]

\[ S \] colludes with last \( k \) parties:

Repeatedly:

- run protocol on choice of \( z_{n-k+1}, \ldots, z_n \)
- learn

\[ f(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n) \]

standard: single evaluation of \( f \)
here: multiple evaluations of \( f \)
security: inherent leakage

\[ f(x_1, x_2, x_3, z_4), \]
\[ f(x_1, x_2, x_3, z'_4), \]
\[ f(x_1, x_2, x_3, z''_4), \]
\[ \ldots \]

\[ f(x_1, \ldots, x_{n-k}, \star) \]

Repeatedly:
- run protocol on choice of \( z_{n-k+1}, \ldots, z_n \)
- learn \( f(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n) \)

standard: single evaluation of \( f \)
here: multiple evaluations of \( f \)
security: inherent leakage

\[ f(x_1, x_2, x_3, z_4), \]
\[ f(x_1, x_2, x_3, z'_4), \]
\[ f(x_1, x_2, x_3, z''_4), \]
\[ \ldots \]

\( S \) colludes with last \( k \) parties:
\[ \Rightarrow \text{adversary gets oracle} \]
\[ f(x_1, \ldots, x_{n-k}, \star) \]
previous work

Q. what can we compute with secure, one-pass protocols? [HLPI11]

✓ sum, selection, symmetric functions e.g. majority

  (via practical protocols)

✗ pseudo-random functions
previous work

Q. what can we compute with secure, one-pass protocols? [HLPI1]

✓ sum, selection, symmetric functions e.g. majority

(via practical protocols)

✗ pseudo-random functions

NB. similar models, but no inherent leakage

— more than one pass [SYY99, IKOPS01, AJLTW12]
— non-colluding server [IKYZ09]
previous work

Q. what can we compute with secure, one-pass protocols? [HLPII]

✓ sum, selection, symmetric functions e.g. majority

   (via practical protocols)

× pseudo-random functions

NB. related techniques, different context [IP07, HIK07]
this work

**Theorem.** Secure one-pass protocols for

1. Sparse multi-variate polynomials (DCR)
2. Read-once branching programs (DCR, DDH/DLIN, ...)

this work

**Theorem.** Secure one-pass protocols for

1. Sparse multi-variante polynomials
2. Read-once branching programs

sum

selection, symmetric functions
this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs

- low-degree polynomials
  - e.g. variance

- string matching, finite automata, classification, second-price auction
this work

**Theorem.** Secure one-pass protocols for

1. Sparse multi-variate polynomials
2. Read-once branching programs

“Are at least 3 of \( \{x_1, \ldots, x_4\} \) equal to 1?”
this work

**Theorem.** Secure one-pass protocols for

1. Sparse multi-variate polynomials
2. Read-once branching programs

“Are at least 3 of \(\{x_1, \ldots, x_4\} \) equal to 1?”
this work

**Theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs

\[ f(0, 1, 0, 1) = 0 \]

"are at least 3 of \( \{x_1, \ldots, x_4\} \) equal to 1?"
this work

**Theorem.** Secure one-pass protocols for

1. Sparse multi-variate polynomials
2. Read-once branching programs

**Our protocol.** In public key model

- Right-to-left [IP07] + nested encryption [HLP11]
this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs

**our protocol.** in public key model

- right-to-left [IP07] + nested encryption [HLP11]
- this talk: honest-but-curious (malicious via NIZK / GS proofs)
our protocol (warm-up)
our protocol (warm-up)
our protocol (warm-up)
our protocol (warm-up)

\[
E_1(E_2(E_3(E_4(E_{s}(0))))))\]

\[
P_1 x_1 = 0 \quad P_2 x_2 = 1
\]
our protocol (warm-up)

\[ E_1 \left( E_2 \left( E_3 \left( E_4 \left( E_s(0) \right) \right) \right) \right) \]

\[ E_1 \left( E_2 \left( E_3 \left( E_4 \left( E_s(1) \right) \right) \right) \right) \]

\[ P_1 \]

\[ x_1 = 0 \]

\[ P_2 \]

\[ x_2 = 1 \]
our protocol (warm-up)

\[ P_1 \times 1 = 0 \]
\[ P_2 \times 2 = 1 \]
\[ P_3 \times 3 = 0 \]
our protocol (warm-up)
our protocol (warm-up)
our protocol (warm-up)
our protocol (warm-up)

\[
E_1(E_2(E_3(E_4(E_5(0))))))
\]
\[
E_1(E_2(E_3(E_4(E_5(1))))))
\]

\[
P_1 \begin{array}{c} x_1 = 0 \end{array}
\]
\[
P_2 \begin{array}{c} x_2 = 1 \end{array}
\]
\[
P_3 \begin{array}{c} x_3 = 0 \end{array}
\]
\[
P_4 \begin{array}{c} x_4 = 1 \end{array}
\]

next. propagate encrypted node labels “homomorphically”
our protocol (warm-up)

\[ E_s(0) \]

\[ E_1(E_2(E_3(E_4(E_5(0))))) \]
\[ E_1(E_2(E_3(E_4(E_5(1))))) \]

next. propagate encrypted node labels “homomorphically”
our protocol
our protocol

\[
P_1
\]

\[x_1 = 0\]
our protocol

\[
\begin{align*}
E_1(E_2(E_3(E_4(E_5(0)))))) \\
E_1(E_2(E_3(E_4(E_5(1))))))
\end{align*}
\]
our protocol

\[ E_1(E_2(E_3(E_4(E_s(0)))))) \]
\[ E_1(E_2(E_3(E_4(E_s(1)))))) \]

\[ x_1 = 0 \]
our protocol

\[
E_2(E_3(E_4(E_s(0)))) \\
E_2(E_3(E_4(E_s(0)))) \\
E_2(E_3(E_4(E_s(1))))
\]

\[
P_1 \\
x_1 = 0
\]
our protocol

\[
E_2(E_3(E_4(E_s(0))))
\]

\[
E_2(E_3(E_4(E_s(0))))
\]

\[
E_2(E_3(E_4(E_s(1))))
\]

\[
x_1 = 0
\]
our protocol

\[\begin{align*}
E_2(E_3(E_4(E_s(0)))) \\
E_2(E_3(E_4(E_s(0)))) \\
E_2(E_3(E_4(E_s(1)))) \\
\end{align*}\]

\[x_2 = 1\]
our protocol

\[ E_2(E_3(E_4(E_s(0)))) \]
\[ E_2(E_3(E_4(E_s(0)))) \]
\[ E_2(E_3(E_4(E_s(1)))) \]

\[ x_2 = 1 \]
our protocol

\[
\begin{align*}
E_1(E_2(E_3(E_4(E_s(0))))) \\
E_2(E_3(E_4(E_s(0)))) \\
E_2(E_3(E_4(E_s(0)))) \\
E_3(E_4(E_s(0))) \\
E_3(E_4(E_s(1))) \\
x_2 = 1
\end{align*}
\]
our protocol

\[
E_3(E_4(E_s(0)))
\]

\[
E_3(E_4(E_s(0)))
\]

\[
E_3(E_4(E_s(1)))
\]

\[
x_2 = 1
\]
our protocol

\[
E_3(E_4(E_s(0)))
\]

\[
E_3(E_4(E_s(0)))
\]

\[
E_3(E_4(E_s(1)))
\]

\[
x_3 = 0
\]
our protocol

\[ E_3(E_4(E_s(0))) \]
\[ E_3(E_4(E_s(0))) \]
\[ E_3(E_4(E_s(1))) \]

result = 0

\( P_2 \times 1 = 0 \)
\( P_2 \times 2 = 1 \)
\( P_4 \times 4 = 1 \)

\( S \)

\[ x_3 = 0 \]
our protocol

\[ P_3 \]

\[ x_3 = 0 \]

\[ E_4(E_s(0)) \]

\[ E_4(E_s(0)) \]
our protocol

\[ E_1 \left( E_2 \left( E_3 \left( E_4 \left( E_s(0) \right) \right) \right) \right) \]

\[ E_1 \left( E_2 \left( E_3 \left( E_4 \left( E_s(0) \right) \right) \right) \right) \]

\[ E_2 \left( E_3 \left( E_4 \left( E_s(0) \right) \right) \right) \]

\[ E_3 \left( E_4 \left( E_s(0) \right) \right) \]

\[ E_4 \left( E_s(0) \right) \]

\[ P_1 \times 1 = 0 \]

\[ P_2 \times 2 = 1 \]

\[ P_3 \times 3 = 0 \]

\[ P_4 \times 4 = 1 \]

\[ x_3 = 0 \]
our protocol

\[
\begin{align*}
E_1(E_2(E_3(E_4(E_s(0))))) \\
E_2(E_3(E_4(E_s(0)))) \\
E_2(E_3(E_4(E_s(1)))) \\
E_3(E_4(E_s(0))) \\
E_3(E_4(E_s(1))) \\
E_4(E_s(0)) \\
E_4(E_s(1)) \\
P_2 \times 1 = 0 \\
P_2 \times 2 = 1 \\
P_3 \times 3 = 0 \\
P_4 \times 4 = 1 \\
x_4 = 1
\end{align*}
\]
our protocol

\[
E_1(E_2(E_3(E_4(E_s(0))))),
\]

\[
E_1(E_2(E_3(E_4(E_s(0))))),
\]

\[
E_1(E_2(E_3(E_4(E_s(1))))),
\]

\[
E_2(E_3(E_4(E_s(0)))),
\]

\[
E_2(E_3(E_4(E_s(0)))),
\]

\[
E_2(E_3(E_4(E_s(0)))),
\]

\[
E_2(E_3(E_4(E_s(1)))),
\]

\[
E_3(E_4(E_s(0))),
\]

\[
E_3(E_4(E_s(0))),
\]

\[
E_3(E_4(E_s(0))),
\]

\[
E_4(E_s(0))
\]

\[
E_4(E_s(0))
\]

\[
S
\]

\[
P_4
\]

\[
x_4 = 1
\]
our protocol

\[
E_1(E_2(E_3(E_4(E_{s(0)}))))
\]

\[
P_2 \neq S \quad \text{and} \quad P_2 \neq x_1 \quad \text{but not} \quad x_1 = 1
\]
our protocol

\[ E_1(E_2(E_3(E_4(E_s(0)))))) \]

\[ E_1(E_2(E_3(E_4(E_s(1)))))) \]

\[ E_2(E_3(E_4(E_s(0)))) \]

\[ E_3(E_4(E_s(0))) \]

\[ E_4(E_s(0)) \]

\[ P_2 \]

\[ P_1 \]

\[ P_2 \]

\[ P_3 \]

\[ P_4 \]

\[ x_4 = 1 \]
our protocol

\[
\begin{align*}
\mathcal{E}_1 & (\mathcal{E}_2 (\mathcal{E}_3 (\mathcal{E}_4 (\mathcal{E}_s(0)))))) \\
\mathcal{E}_1 & (\mathcal{E}_2 (\mathcal{E}_3 (\mathcal{E}_4 (\mathcal{E}_s(0)))))) \\
\mathcal{E}_2 & (\mathcal{E}_3 (\mathcal{E}_4 (\mathcal{E}_s(0)))) \\
\mathcal{E}_3 & (\mathcal{E}_4 (\mathcal{E}_s(0))) \\
\mathcal{E}_4 & (\mathcal{E}_s(0))
\end{align*}
\]

result = 0

\[P_1 \times 1 = 0, \quad P_2 \times 2 = 1, \quad P_3 \times 3 = 0, \quad P_4 \times 4 = 1\]

\[E_s(0)\]

\[S\]
our protocol

\[
E_1(E_2(E_3(E_4(E_s(0))))),
\]

\[
E_1(E_2(E_3(E_4(E_s(1))))),
\]

\[
E_1(E_2(E_3(E_s(0)))),
\]

\[
E_1(E_2(E_s(0))),
\]

\[
P_2 \not= S_1 \rightarrow P_3 \not= S_2 \rightarrow P_4 \not= S_3.
\]

\[
\text{efficiency. } O(\text{width}) \text{ exponentiations per player under DCR, DDH/DLIN, ...}
\]
our protocol

\[
E_1(E_2(E_3(E_4(E_{s}(0))))),
\]

\[
E_2(E_3(E_4(E_{s}(0)))),
\]

\[
E_3(E_4(E_{s}(0))),
\]

\[
E_4(E_{s}(0)).
\]

**efficiency.** \(O(\text{width})\) exponentiations per player under DCR, DDH/DLIN, ...

**security I.** honest \(S\) – all messages protected by \(E_s(\cdot)\)
our protocol

\[
E_1(E_2(E_3(E_4(E_{s}(0)))))
\]

\[
E_1(E_2(E_3(E_4(E_{s}(0))))
\]

\[
E_2(E_3(E_4(E_{s}(0))))
\]

\[
E_3(E_4(E_{s}(0)))
\]

\[
E_4(E_{s}(0))
\]

\[
P_2 ! S
\]

\[
P_1 x 1 = 0
\]

\[
P_2 x 2 = 1
\]

\[
P_3 x 3 = 0
\]

\[
P_4 x 4 = 1
\]

result = 0

**efficiency.** $O(\text{width})$ exponentiations per player under DCR, DDH/DLIN, ...

**security I.** honest $S$ – all messages protected by $E_s(\cdot)$

**security II.** corrupt $S, P_3, P_4$ – need to simulate view given $f(x_1, x_2, \star)$

but not $x_1, x_2$
our protocol

**efficiency.** \( O(\text{width}) \) exponentiations per player under DCR, DDH/DLIN, ...

**security I.** honest \( S \) – all messages protected by \( E_s(\cdot) \)

**security II.** corrupt \( S, P_3, P_4 \) – need to simulate view given \( f(x_1, x_2, \star) \)

but not \( x_1, x_2 \)
our protocol

“How to simulate these node labels (unencrypted)?”

\[
\begin{align*}
E_3(E_4(E_s(0))) \\
E_3(E_4(E_s(0))) \\
E_3(E_4(E_s(1)))
\end{align*}
\]

\[
f(x_1, x_2, \star) \text{ oracle}
\]

\[
sim-view
\]
our protocol

“How to simulate these node labels (unencrypted)?”

- for each node, use BFS to find a path from start node

\[ E_3(E_4(E_5(0))) \]
\[ E_3(E_4(E_5(0))) \]
\[ E_3(E_4(E_5(1))) \]
our protocol

“How to simulate these node labels (unencrypted)?”

- for each node, use BFS to find a path from start node
- call oracle on inputs induced by path

\[ f(x_1, x_2, \star) \] oracle

\[ E_3(E_4(E_5(0))) \]
\[ E_3(E_4(E_5(0))) \]
\[ E_3(E_4(E_5(1))) \]
conclusion

**this work.** secure one-pass protocols

1. sparse multi-variate polynomials
2. read-once branching programs

**open questions.**

- larger classes, e.g. linear branching programs [HIK07]?
- impossibility results / complete characterization?
- better efficiency, e.g. second-price auctions?
the end