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

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

\[ x_1 \quad P_1 \]
\[ x_2 \quad P_2 \]
\[ x_3 \quad P_3 \]
\[ \ldots \]
\[ x_n \quad P_n \]

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

```
x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_n
\quad P_1 \quad P_2 \quad P_3 \quad \ldots \quad P_n
```

- **electronic voting**
- **secure auctions**

**“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 \textbf{once} with server in fixed order
- server announces result
- server may be corrupt and \textbf{colluding} with parties
  \( \Rightarrow \) new technical challenge beyond standard MPC
security: inherent leakage

$S$ colludes with last $k$ parties:
security: inherent leakage

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

\[ S \text{ colludes with last } k \text{ parties:} \]

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

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

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

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

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

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

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

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

\[ \text{sim-view} \]
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
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

**NB.** similar models, but no inherent leakage

— more than one pass [SYY99, IKOPS01, AJLTVM12]
— non-colluding server [IKYZ09]
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

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-variate polynomials
2. Read-once branching programs

1. Sum

2. 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

```
\begin{align*}
\text{\texttt{x}_4} & \rightarrow 0 \\
\text{\texttt{x}_3} & \rightarrow 0 \\
\text{\texttt{x}_2} & \rightarrow 0 \\
\text{\texttt{x}_1} & \rightarrow 0 \\
\text{0} & \\
\text{l} &
\end{align*}
```

"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

![Diagram](image)

**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)

\[
P_1 \\
x_1 = 0
\]
our protocol (warm-up)

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

\[P_1 \times 1 = 0\]

\[P_2 \times 2 = 1\]

\[P_3 \times 3 = 0\]

\[P_4 \times 4 = 1\]

\[x_1 = 0\]
our protocol (warm-up)

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

\[ P_1 \times 1 = 0 \]

\[ P_2 \times 2 = 1 \]

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

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

\[ P_1 x_1 = 0 \]
\[ P_2 x_2 = 1 \]
\[ P_3 x_3 = 0 \]
our protocol (warm-up)
our protocol (warm-up)
our protocol (warm-up)

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

\[ P_1 \times 1 = 0 \]
\[ P_2 \times 2 = 1 \]
\[ P_3 \times 3 = 0 \]
\[ P_4 \times 4 = 1 \]
our protocol (warm-up)

next. propagate encrypted node labels “homomorphically”

\[ E_1(E_2(E_3(E_4(E_5(0)))))) \]
\[ E_1(E_2(E_3(E_4(E_5(1)))))) \]
our protocol (warm-up)

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

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

\[
P_2 \neq \text{S}_{\text{honest} - \text{all messages protected by } E_{s(0)}}
\]

\[
P_3 \neq \text{P}_{\text{corrupt} \text{; } P_3; P_4 - \text{need to simulate view given } f(x_1; x_2; \cdots)}
\]

\[
x_1 = 0
\]
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

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

\[P_1\]

\[x_1 = 0\]
our protocol

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

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

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

\[
P_1
\]

\[
x_1 = 0
\]
our protocol

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

result = 0

efficiency.

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

security II.
corrupt S; P_3; P_4 – need to simulate view given \(f(x_1; x_2; \star)\) but not \(x_1; x_2\).

\(x_1 = 0\)

\(P_1\)

S
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

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

\[
P_2 \\
x_2 = 1
\]

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

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

\[
P_2
\]

\[
x_2 = 1
\]
our protocol

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

\[x_2 = 1\]
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

\[ P_3 \]

\[ x_3 = 0 \]

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

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

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

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

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

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

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

\[
E_4(E_s(0))
\]

\[
P_1 = 0
\]

\[
P_2 = 1
\]

\[
P_3 = 0
\]

\[
P_4 = 1
\]

\[
x_3 = 0
\]

result = 0

security I. honest S – all messages protected by E_s.

security II. corrupt S, P_3, P_4 – need to simulate view given f(x_1; x_2; \cdots), but not x_1; x_2.

efficiency. O(width) exponentiations per player under DCR, DDH/DLIN,...
our protocol

\[ E_1(\mathcal{E}_2(\mathcal{E}_3(\mathcal{E}_4(E_s(0)))) \]

\[ E_2(\mathcal{E}_3(\mathcal{E}_4(E_s(0)))) \]

\[ E_3(\mathcal{E}_4(E_s(0))) \]

\[ E_4(E_s(0)) \]

\[ x_3 = 0 \]
our protocol

\[
\begin{align*}
E_1(E_2(E_3(E_4(E_s(0))))), \\
E_2(E_3(E_4(E_s(0)))), \\
E_2(E_3(E_4(E_s(1)))), \\
E_4(E_s(0)), \\
P_2 x_1 = 0, \\
P_2 x_2 = 1, \\
P_3 x_3 = 0, \\
P_4 x_4 = 1,
\end{align*}
\]

result = 0

efficiency.

O(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 $x_1, x_2$.  

\[x_4 = 1\]
our protocol

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

\[x_4 = 1\]
our protocol

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

result = 0

\[
P_2 \neq S
\]

\[
P_2 \neq S
\]

\[
P_3 \neq S
\]

\[
P_4 \neq S
\]

\[
x_4 = 1
\]
our protocol

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

\[
P_2 \overset{!}{\rightarrow} S \quad P_3 \overset{x_1}{\rightarrow} P_4 \overset{x_2}{\rightarrow} S.
\]

result = 0

\[
x_4 = 1
\]

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

security I. honest \(S\) – all messages protected by \(E_s(0)\).

security II. corrupt \(P_3, P_4\) – need to simulate view given \(f(x_1, x_2, \ast)\) but not \(x_1, x_2\).
our protocol

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

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

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

result = 0

\[S\]
our protocol

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

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

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

**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 \(f(x_1, x_2, *)\)

but not \(x_1, x_2\)
our protocol

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

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

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

\[ \text{simulator} \]

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

\( E_3(E_4(E_5(0))) \)
\( E_3(E_4(E_5(0))) \)
\( E_3(E_4(E_5(1))) \)

simulator

\( \sim \text{-view} \)
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