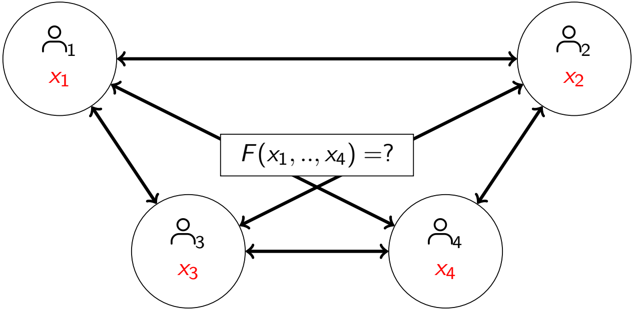
Secure Computations on Shared Polynomials and Application to Private Set Operations

Lucas Ottow

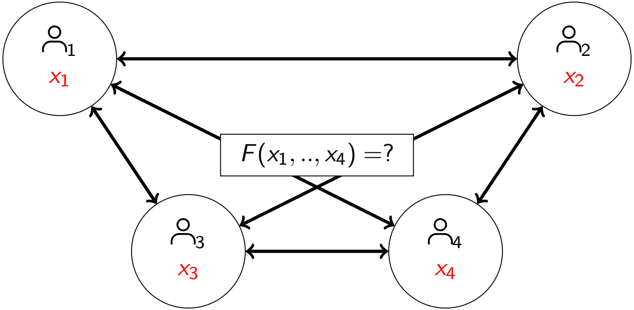
Pascal Giorgi[1], Fabien Laguillaumie[1], **Lucas Ottow**[1,2], Damien Vergnaud[2]

[1] LIRMM, UM, Montpellier, France
[2] LIP6, Sorbonne Université, Paris, France

Multiparty Computation

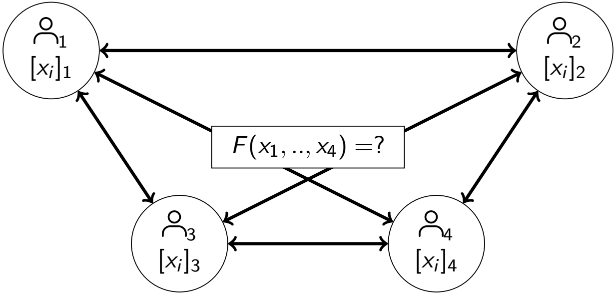


Multiparty Computation

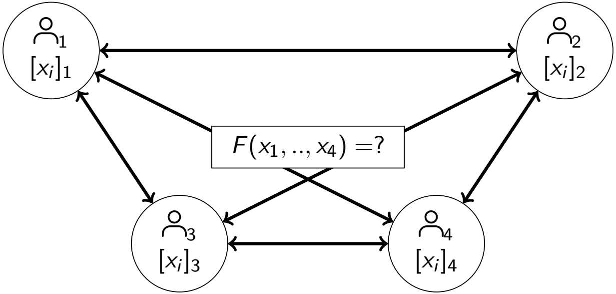


- General result by Yao ('82)

Secret Sharing in MPC

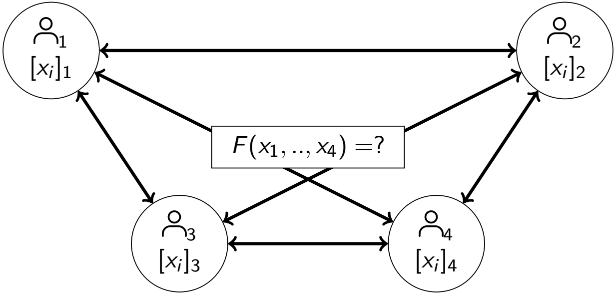


Secret Sharing in MPC



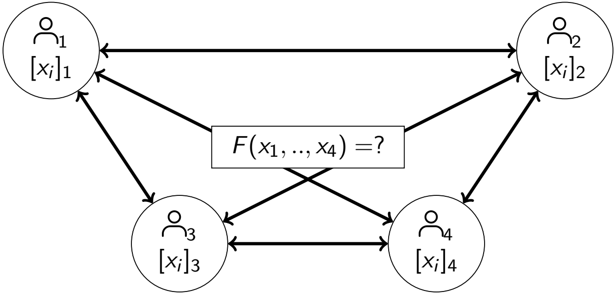
- Sol: share secret input + compute jointly : Beaver89, BGW88.

Secret Sharing in MPC



- Sol: share secret input + compute jointly : Beaver89, BGW88.
- Costly operation: product of two shared elements

Secret Sharing in MPC



- Sol: share secret input + compute jointly : Beaver89, BGW88.
- Costly operation: product of two shared elements
- Our goal: less secure multiplications (sec. mult.) + constant round

How to compute on shares

- Linear sharing over $\mathbb{F}_q$: $[x] \implies [x]_1 + ... + [x]_m = x$

How to compute on shares

- Linear sharing over $\mathbb{F}_q$: $[x] \implies [x]_1 + ... + [x]_m = x$
- Base operations:

How to compute on shares

- Linear sharing over $\mathbb{F}_q$: $[x] \implies [x]_1 + ... + [x]_m = x$
- Base operations:
  - Compute $[a+b]$ given $[a]$ and $[b]$: $[a+b]_j = [a]_j + [b]_j$.
  - Compute $[ca]$ given $[a]$ and c public: $[ca]_j = c \times [a]_j$.

How to compute on shares

- Linear sharing over $\mathbb{F}_q$: $[x] \implies [x]_1 + ... + [x]_m = x$
- Base operations:
  - Compute $[a+b]$ given $[a]$ and $[b]$: $[a+b]_j = [a]_j + [b]_j$.
  - Compute $[ca]$ given $[a]$ and c public: $[ca]_j = c \times [a]_j$.
  - Compute $[ab]$ given $[a], [b]$: requires communications.

How to compute on shares

- Linear sharing over $\mathbb{F}_q$: $[x] \implies [x]_1 + \ldots + [x]_m = x$
- Base operations:
  - Compute $[a+b]$ given $[a]$ and $[b]$: $[a+b]_j = [a]_j + [b]_j$.
  - Compute $[ca]$ given $[a]$ and c public: $[ca]_j = c \times [a]_j$.
  - Compute $[ab]$ given $[a], [b]$: requires communications.
- Sharing over $\mathbb{F}_q[X]$: $f = \sum f_i X^i$: $([f_0], \ldots, [f_{d-1}])$

How to compute on shares

- Linear sharing over $\mathbb{F}_q$: $[x] \implies [x]_1 + ... + [x]_m = x$
- Base operations:
  - Compute $[a+b]$ given $[a]$ and $[b]$: $[a+b]_j = [a]_j + [b]_j$.
  - Compute $[ca]$ given $[a]$ and c public: $[ca]_j = c \times [a]_j$.
  - Compute $[ab]$ given $[a], [b]$: requires communications.
- Sharing over $\mathbb{F}_q[X]$: $f = \sum f_i X^i$: $([f_0], ..., [f_{d-1}])$
  - shared product?

How to compute on shares

- Linear sharing over $\mathbb{F}_q$: $[x] \implies [x]_1 + ... + [x]_m = x$
- Base operations:
  - Compute $[a+b]$ given $[a]$ and $[b]$: $[a+b]_j = [a]_j + [b]_j$.
  - Compute $[ca]$ given $[a]$ and c public: $[ca]_j = c \times [a]_j$.
  - Compute $[ab]$ given $[a], [b]$: requires communications.
- Sharing over $\mathbb{F}_q[X]$: $f = \sum f_i X^i$: $([f_0], ..., [f_{d-1}])$
  - shared product?
    Naive method: $[fg] = ([f_0], ..., [f_{d-1}]) * ([g_0], ..., [g_{d-1}])$.
    $\mathcal{O}(d^2)$ secure multiplication total.

How to compute on shares

- Linear sharing over $\mathbb{F}_q$: $[x] \implies [x]_1 + ... + [x]_m = x$
- Base operations:
  - Compute $[a+b]$ given $[a]$ and $[b]$: $[a+b]_j = [a]_j + [b]_j$.
  - Compute $[ca]$ given $[a]$ and c public: $[ca]_j = c \times [a]_j$.
  - Compute $[ab]$ given $[a], [b]$: requires communications.
- Sharing over $\mathbb{F}_q[X]$: $f = \sum f_i X^i$: $([f_0], ..., [f_{d-1}])$
  - shared product?
    Naive method: $[fg] = ([f_0], ..., [f_{d-1}]) * ([g_0], ..., [g_{d-1}])$.
    $\mathcal{O}(d^2)$ secure multiplication total.
  - Mohassel, Franklin (PKC'06) do it better: $\mathcal{O}(d)$ sec. mult.

Previous work and our contribution

- Mohassel, Franklin (PKC'06): operations on shared polynomial (multiplication, division, interpolation...)

Previous work and our contribution

- Mohassel, Franklin (PKC'06): operations on shared polynomial (multiplication, division, interpolation...)
- Our results: (τ is a constant)

Previous work and our contribution

- Mohassel, Franklin (PKC'06): operations on shared polynomial (multiplication, division, interpolation...)
- Our results: (τ is a constant)
  - Multiplication of n polynomials of degree $< d$:

$$\mathcal{O}(n^2 d) \rightarrow \mathcal{O}(\tau n^{1+1/\tau} d)$$

Previous work and our contribution

- Mohassel, Franklin (PKC'06): operations on shared polynomial (multiplication, division, interpolation...)

- Our results: (τ is a constant)
  - Multiplication of n polynomials of degree $< d$:

$$\mathcal{O}(n^2 d) \to \mathcal{O}(\tau n^{1+1/\tau} d)$$

  - Interpolation and multipoint evaluation of polynomials of degree $< d$:

$$\mathcal{O}(d^2) \to \mathcal{O}(\tau d^{1+1/\tau})$$

Previous work and our contribution

- Mohassel, Franklin (PKC'06): operations on shared polynomial (multiplication, division, interpolation...)

- Our results: (τ is a constant)
  - Multiplication of n polynomials of degree $< d$:

$$\mathcal{O}(n^2 d) \to \mathcal{O}(\tau n^{1+1/\tau} d)$$

  - Interpolation and multipoint evaluation of polynomials of degree $< d$:

$$\mathcal{O}(d^2) \to \mathcal{O}(\tau d^{1+1/\tau})$$

  - Application to privacy preserving set operations.

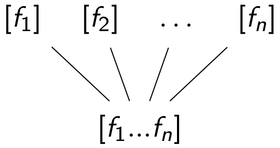
Our method: an example

Op: $[f_1], ..., [f_n]$ (deg $< d$) $\longrightarrow$ $[f_1 ... f_n]$

Our method: an example

<u>Op:</u> $[f_1], \dots, [f_n]$ (deg $< d$) $\longrightarrow$ $[f_1 \dots f_n]$

Mohassel, Franklin (PKC'06)



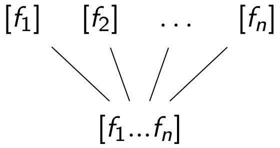
n poly at a time

$1 \times \mathcal{O}(n^2 d)$ sec. mult

Our method: an example

Op: $[f_1], \dots, [f_n]$ $(\deg < d)$ $\longrightarrow$ $[f_1 \dots f_n]$

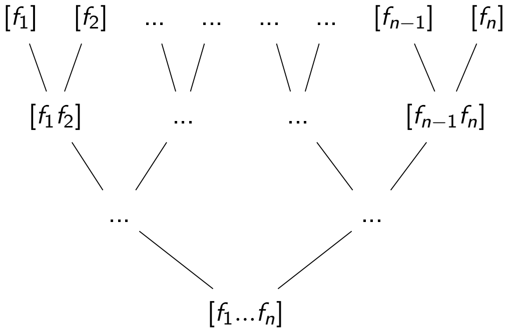
Mohassel, Franklin (PKC'06)



n poly at a time

$1 \times \mathcal{O}(n^2 d)$ sec. mult

Binary tree



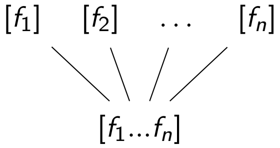
2 poly at a time

$\log n \times \mathcal{O}(nd)$ sec. mult

Our method: an example

Op: $[f_1], \ldots, [f_n]$ (deg $< d$) $\longrightarrow$ $[f_1 \ldots f_n]$

Mohassel, Franklin (PKC'06)

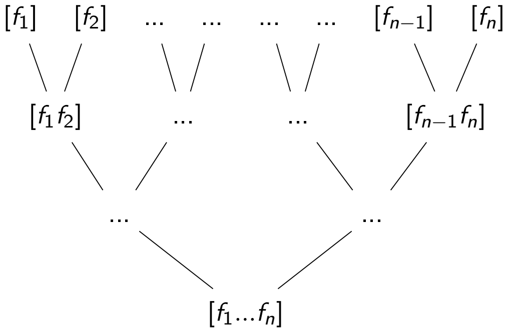


n poly at a time

$1 \times \mathcal{O}(n^2 d)$ sec. mult

$\mathcal{O}(1)$ rounds

Binary tree



2 poly at a time

$\log n \times \mathcal{O}(nd)$ sec. mult

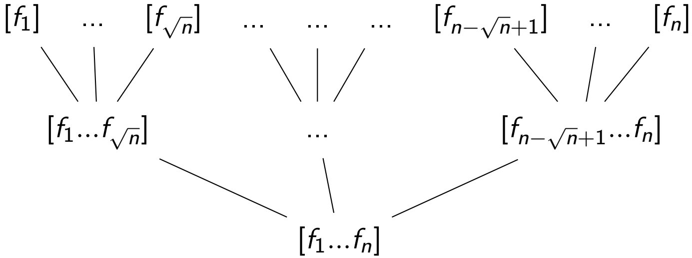
$\mathcal{O}(\log n)$ rounds

Our method: an example

"Squished" tree

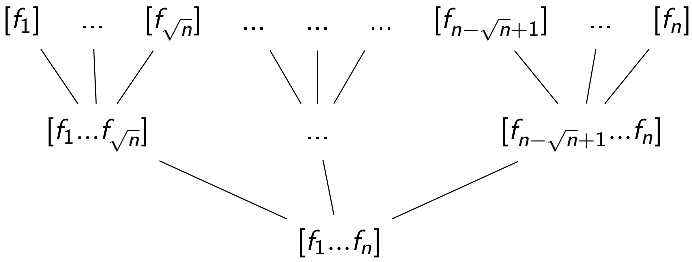
Our method: an example

<u>"Squished" tree</u>



Our method: an example

"Squished" tree

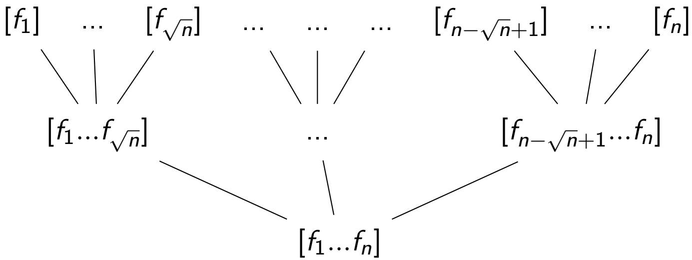


$\sqrt{n}$ poly at a time

$\sqrt{n} \times \mathcal{O}((\sqrt{n})^2 d)$ sec. mult.

Our method: an example

"Squished" tree



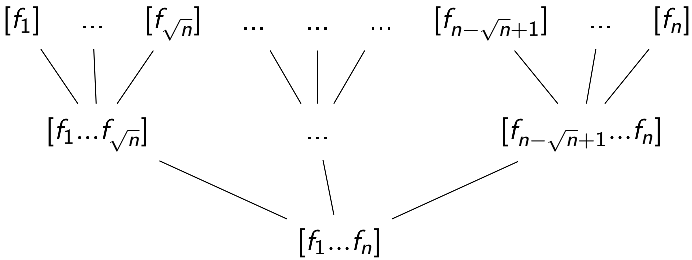
$\sqrt{n}$ poly at a time

$\sqrt{n} \times \mathcal{O}((\sqrt{n})^2 d)$ sec. mult.

$\mathcal{O}(1)$ rounds (2 steps)

Our method: an example

"Squished" tree



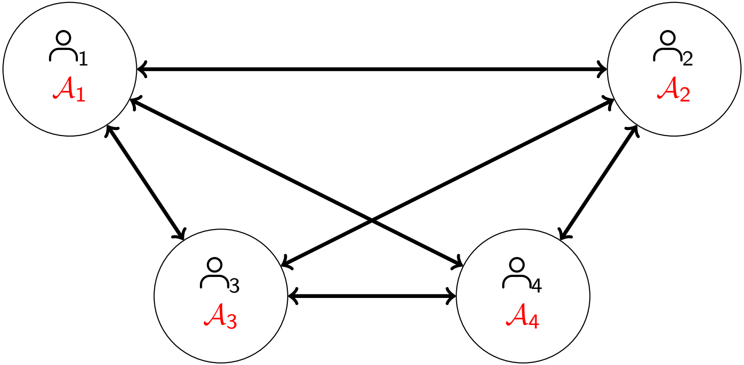
$\sqrt{n}$ poly at a time

$\sqrt{n} \times \mathcal{O}((\sqrt{n})^2 d)$ sec. mult.

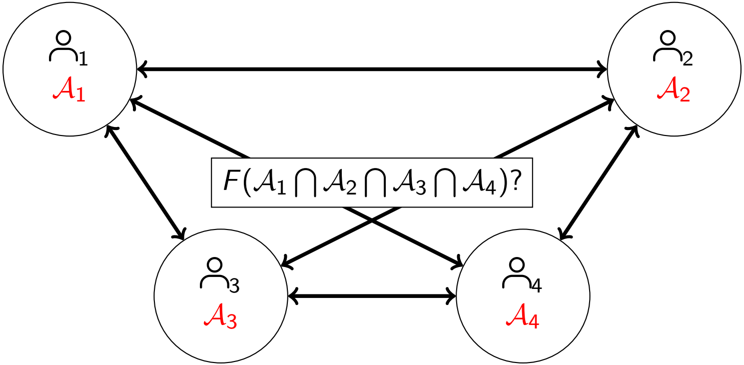
$\mathcal{O}(1)$ rounds (2 steps)

- Generalization: $\mathcal{O}(\tau)$ rounds and $\mathcal{O}(\tau n^{1+1/\tau} d)$ sec. mult.

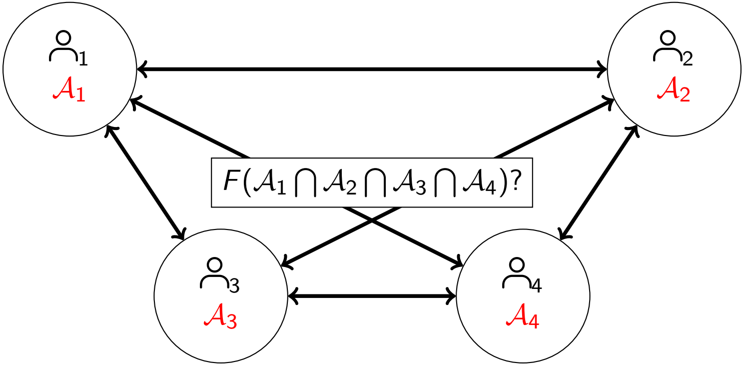
Privacy Preserving operations



Privacy Preserving operations



Privacy Preserving operations



- Example: social network, investigations, ...

Privacy Preserving operations

- Main idea: $\mathcal{A}_j$ represented as shared polynomials: $P_j = \prod_{\alpha \in \mathcal{A}_j}(X - \alpha)$.

Privacy Preserving operations

- Main idea: $\mathcal{A}_j$ represented as shared polynomials: $P_j = \prod_{\alpha \in \mathcal{A}_j} (X - \alpha)$.
- Use our evaluation techniques.

Privacy Preserving operations

- Main idea: $\mathcal{A}_j$ represented as shared polynomials: $P_j = \prod_{\alpha \in \mathcal{A}_j} (X - \alpha)$.
- Use our evaluation techniques.
- 1st protocol: probabilistic Private Disjointess Test/PSI-emptiness, $\mathcal{O}(mn + \tau n^{1+1/\tau})$

Privacy Preserving operations

- Main idea: $\mathcal{A}_j$ represented as shared polynomials: $P_j = \prod_{\alpha \in \mathcal{A}_j}(X - \alpha)$.
- Use our evaluation techniques.
- 1st protocol: probabilistic Private Disjointess Test/PSI-emptiness, $\mathcal{O}(mn + \tau n^{1+1/\tau})$
- 2nd protocol: Many other problems without error in $\mathcal{O}(mn \log \log q + \tau mn^{1+1/\tau})$ (using techniques from Damgård et al. (TCC'06))

Future work

- Other operations on polynomials: gcd is currently costly and many applications.

Future work

- Other operations on polynomials: gcd is currently costly and many applications.
- Other applications to cryptographic problems

Future work

- Other operations on polynomials: gcd is currently costly and many applications.
- Other applications to cryptographic problems
- Operations on polynomial matrices.

Future work

- Other operations on polynomials: gcd is currently costly and many applications.
- Other applications to cryptographic problems
- Operations on polynomial matrices.

Thank you for your attention!