MPC in the Preprocessing Model

A Brief Tutorial

Peter Scholl

IISc Bangalore, 19 February 2020

AARHUS
UNIVERSITY

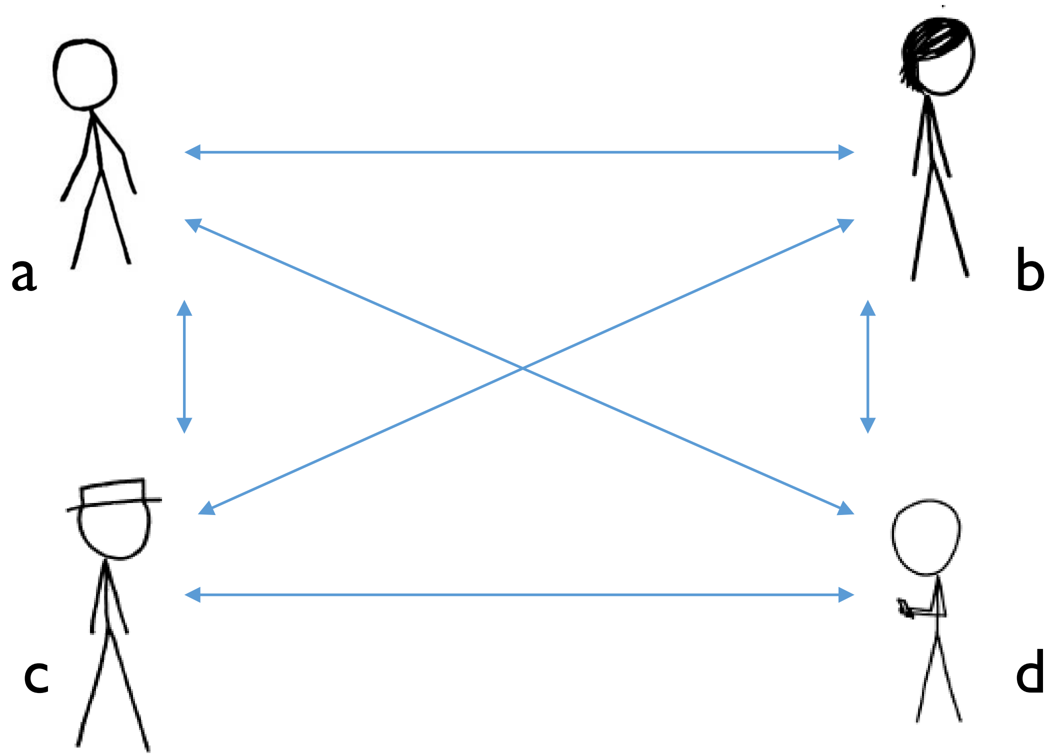
Background: dishonest majority MPC

- Up to $t = n - 1$ parties may be corrupt
- In this setting:
  - Can compute any function with computational assumptions
  - Must settle for security with abort and unfairness
  - Can't have unconditional security ☹
- In the preprocessing model:
  - $t = n - 1$ and unconditional security possible ☺ (with abort)
- Today: only static corruptions

Outline

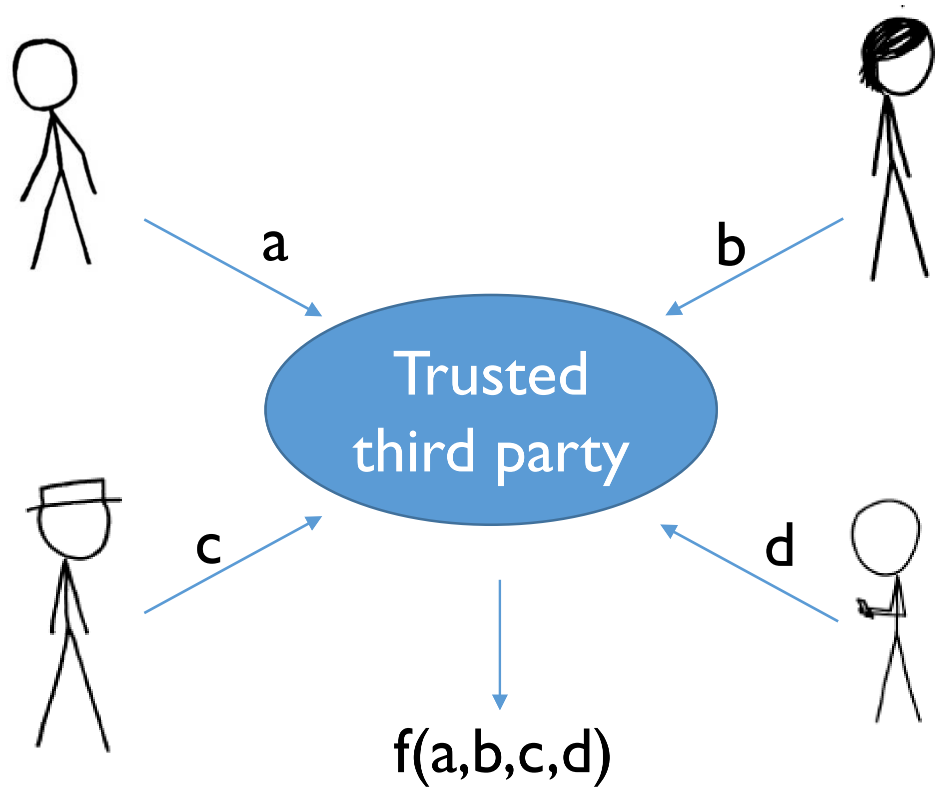
- Warm-up: One-time truth tables (2PC, passive security)
- MPC for arithmetic circuits (passive "GMW protocol")
- Active security with information-theoretic MACs
    - ➢Pairwise MACs ("BDOZ" or "TinyOT" style)
    - ➢Global MACs ("SPDZ" style)

Secure Multi-Party Computation



Goal: Compute f(a,b,c,d)

MPC should be as good as using a trusted third party



MPC in the preprocessing model

- Preprocessing can be done in advance, before inputs known
- Online phase:
  - After inputs are known
  - Lightweight: only constant factor slower than plaintext, in some cases

Where does the preprocessing come from?

- This talk: imagine a "trusted dealer"

- In practice:
  - Use a protocol based on e.g. OT or HE (more on Weds.)
  - Non-colluding 3rd party
  - Trusted hardware device

Warm-up: One time truth table protocol

[Ishai Kushilevitz Meldgaard Orlandi Paskin, *TCC 13*]

- 2-PC for any function
- Very simple, but inefficient
- Note: can be extended to MPC and active security

One time truth table protocol

1) Take the truth-table of function $f: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$

2) Pick random shifts (r, s) and rotate rows/columns

$s = 1$

$r = 3$

	0	1	2	3
0	1	0	1	1
1	0	0	1	0
2	1	1	0	1
3	1	0	0	1

One time truth table protocol

3) Secret share the permuted truth table:

- Sample random: M_A
- Set $M_B = M_A \oplus$

	0	1	2	3
0	1	0	1	1
1	0	0	1	0
2	1	1	0	1
3	1	0	0	1

One time truth table protocol

M_A, r

Input x

Preprocessing

M_B, s

Input y

$u = x + r \mod 2^n$

$v = y + s$

$M_B[u, v]$

Privacy: one-time pad

Output $M_a[u, v] \oplus M_B[u, v]$

Correctness: from preprocessing

Summary: one-time truth table

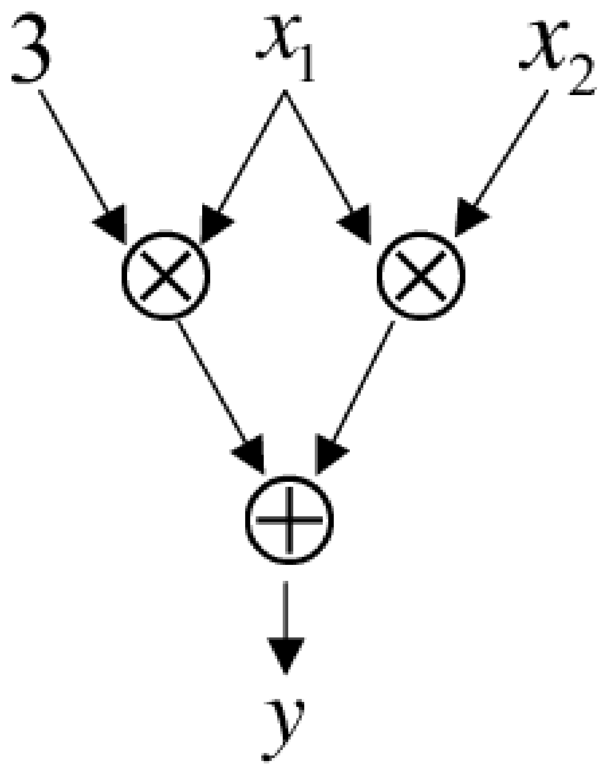
- Optimal communication (|x| + |y|)
- Exponential storage (2^{2n} bits)
- Still useful for small tables
  - E.g. as a building block in larger computations
  - "TinyTable" [DNNR16]

- Warm-up: One-time truth tables (2PC, passive security)
- MPC for arithmetic circuits (passive "GMW protocol")
- Active security with information-theoretic MACs
  - Pairwise MACs ("BDOZ" or "TinyOT" style)
  - Global MACs ("SPDZ" style)

Securely computing arithmetic circuits

- Addition and multiplication gates (over finite field $\boldsymbol{F}$)

- Input + output wires



Main invariant throughout the protocol

- For each wire, with value x, we have:
  - $[x] := (x_1, \ldots, x_n)$
  - $x = x_1 + \cdots + x_n$
  - Party P_i holds $x_i \in F$

Basic operations on $[\cdot]$-shared values

- **Input** x from P_i
  - P_i privately sends random $x_j \in F$ to every other P_j
  - P_i sets $x_i = x - \sum_{j \neq i} x_j$

- **Open** $[x]$
  - Each P_i sends x_i
  - Recover $x = \sum_i x_i$

Basic operations on $[\cdot]$-shared values

- **Linear operation** $[z] := a[x] + b[y]$
  - P_i computes $z_i = a \cdot x_i + b \cdot y_i$

- **Add constant** $[z] := [x] + c$
  - P_1 computes $z_1 = x_1 + c$
  - All other parties let $z_i = x_i$

- N.B. these require no communication

Multiplication of $[x]$ and $[y]$

- Want shares of $z = x \cdot y$
- Observe:

$$x \cdot y = (x + a - a) \cdot (y + b - b)$$

$$= (x + a) \cdot (y + b) - (x + a) \cdot b - a \cdot (y + b) + a \cdot b$$

opened preprocessed

Multiplication of $[x]$ and $[y]$

- Take random, preprocessed triple $[a], [b], [a \cdot b]$
- Open $d = x + a$ and $e = y + b$
- Compute

$$[z] = d \cdot e - d \cdot [b] - e \cdot [a] + [a \cdot b]$$
$$= [x \cdot y]$$

- Warm-up: One-time truth tables (2PC, passive)
- MPC for arithmetic circuits (passive "GMW protocol")
- Active security with information-theoretic MACs
  - ➢Pairwise MACs ("BDOZ" or "TinyOT" style)
  - ➢Global MACs ("SPDZ" style)

What about active security?

- **Problem**: additive secret sharing is not enough
  - Corrupt P_i can send $x_i + e$ during Open
  - Parties reconstruct $x + e$

    $\Rightarrow$ breaks correctness

- **Solution**: use information-theoretic MACs
  - Approach 1: MAC the shares (as in BDOZ or TinyOT)
  - Approach 2: share the MACs (as in SPDZ)

Approach 1: MAC the shares

- $\text{MAC}(x) = \alpha \cdot x + \beta$ in F
  - ➢ Random keys $\alpha, \beta \in F$
  - ➢ Fixed α, fresh β for each MAC

prevents forgery

hides x

- Given $(x, \text{MAC}(x))$, coming up with a MAC on $x' \neq x$ requires guessing α

- MACs are linear

  $\Rightarrow$ can still do linear operations for free

Approach 1: MAC the shares

- MAC each x_i using a key held by P_j:
  - P_i gets x_i and $M_j[x_i] = \mathrm{MAC}(K_j[x_i], x_i)$
  - P_j gets $K_j[x_i] = (\alpha_j, \beta_j[x_i])$

- Modify preprocessing:
  - MAC the triple shares
  - Extra random MAC'd shares, for Input phase

- Check MACs when opening:
  - Send x_i and $M_j[x_i]$ to each P_j to check

Approach 1: MAC the shares

- **Problem**: expensive!
  - $O(n^2)$ MACs for every x
  - Communication and storage now $O(n^2)$ per gate
- Solution: coming up

Approach 2: share the MACs

- MAC the value x, not the share

No β

$$\text{MAC}(x) = \alpha \cdot x$$

- Secret-share the MAC and key α:

$$[x] := (x_1, \ldots, x_n, m_1, \ldots, m_n)$$

- $\mathrm{x} = \sum x_i, \quad \text{MAC}(x) = \sum m_i = \alpha \cdot x$
- P_i has (x_i, m_i)

Approach 2: share the MACs

$$[x] := (x_1, \dots, x_n, m_1, \dots, m_n)$$

$$\mathrm{x} = \sum x_i, \quad \mathrm{MAC}(x) = \sum m_i = \alpha \cdot x$$

Challenge: how to check the MAC without revealing α?

- Parties open $x' = x + e$
- P_i commits to $d_i = \alpha_i \cdot x' - m_i$
  - Note: $d_1 + \cdots + d_n = \alpha \cdot x' - \mathrm{MAC}(x) = \alpha \cdot e$
- Open d_i and check they sum to 0

If $e \neq 0$, have to guess α to pass

MAC the shares vs share the MACs

[BDOZ 11, NNOB 12] [DPSZ 12, DKLPSS 13]

- **Storage:** $O(n^2)$ vs $O(n)$
- **Computation:*** $O(n^2)$ vs $O(n)$
- **Communication:*** $O(n^2)$ vs $O(n)$

(all parties, per gate)

- **MAC check:** 1 round vs 3 rounds

* Assuming delayed batch verification of MACs

Further reading

- General resources: lecture notes, books etc.
    - https://github.com/rdragos/awesome-mpc

- One-time truth tables:
    - *On the Power of Correlated Randomness in Secure Computation* – Ishai, Kushilevitz, Meldgaard, Orlandi, Paskin *(TCC 2013)*

    https://link.springer.com/chapter/10.1007/978-3-642-36594-2_34
    - *Gate-scrambling Revisited - or: The TinyTable protocol for 2-Party Secure Computation* – Damgård, Nielsen, Nielsen, Ranellucci

    https://ia.cr/2016/695

Further reading

- Ciruit-based MPC and active security:
  - "TinyOT": *A New Approach to Practical Active-Secure Two-Party Computation* - Nielsen, Nordholt, Orlandi, Burra (*Crypto 2012)*

  https://ia.cr/2011/091
  - *"BeDOZa": Semi-Homomorphic Encryption and Multiparty Computation* – Bendlin, Damgård, Orlandi, Zakarias (*Eurocrypt 2011)*

  https://ia.cr/2010/514
  - *"SPDZ":*
    - *Multiparty Computation from Somewhat Homomorphic Encryption* – Damgård, Pastro, Smart, Zakarias (*Crypto 2012*)
    - *Practical Covertly Secure MPC for Dishonest Majority – or: Breaking the SPDZ Limits* - Damgård, Keller, Larraia, Pastro, Scholl, Smart *(ESORICS 2013)*

    https://ia.cr/2011/535 https://ia.cr/2012/642