

When it's all just too much: Outsourcing MPC Pre-processing

Peter Scholl, Nigel P. Smart and **Tim Wood**

University of Bristol

12 December 2017

Outline

MPC

- Overview

- Pre-processing Model

SPDZ Family of MPC

- Linear Secret-sharing

- Opening secrets

- Multiplication of secrets

Generating Pre-processed Data

Protocol

Outline

MPC

- Overview

- Pre-processing Model

SPDZ Family of MPC

- Linear Secret-sharing

- Opening secrets

- Multiplication of secrets

Generating Pre-processed Data

Protocol

What is Multi-Party Computation?

What is Multi-Party Computation?

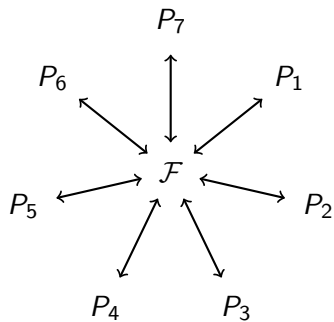
A set of parties computing a function

on their secret input

so that all parties learn the output

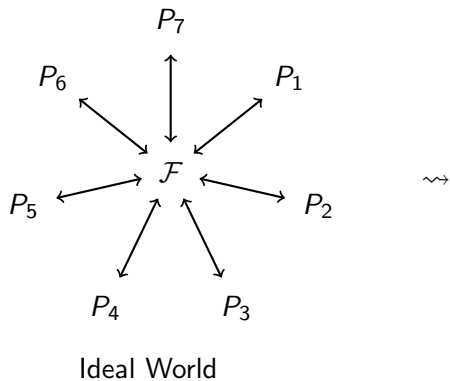
and nothing else about other parties' input that can't be deduced from the output alone.

What is Multi-Party Computation?

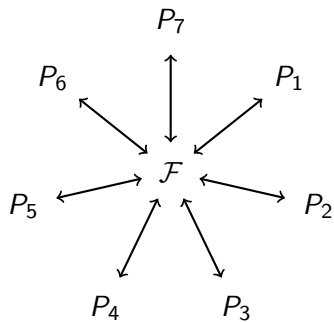


Ideal World

What is Multi-Party Computation?

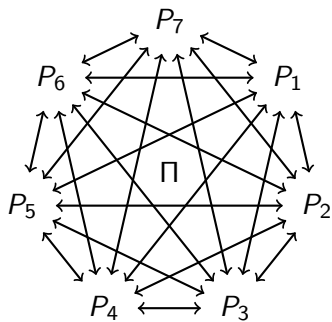


What is Multi-Party Computation?



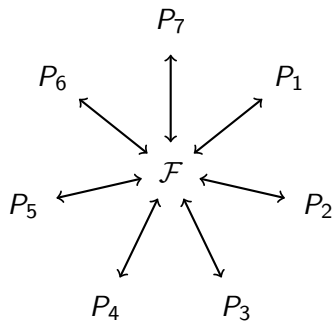
Ideal World

\rightsquigarrow



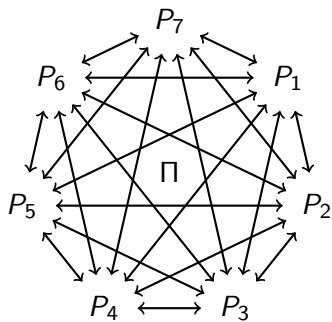
Real World

What is Multi-Party Computation?



Ideal World

\rightsquigarrow



Real World

Possible guarantees:

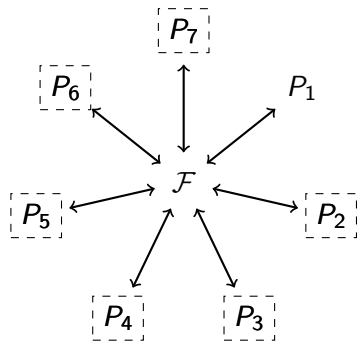
Privacy/Secrecy

Correctness

Fairness

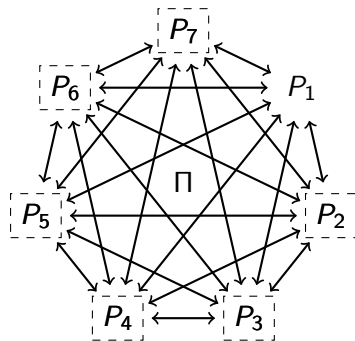
(etc.)

What is Multi-Party Computation?



Ideal World

\rightsquigarrow



Real World

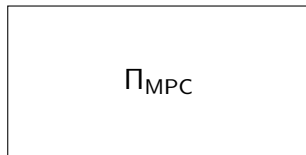
Corruption models:

Active/Passive

Access structure

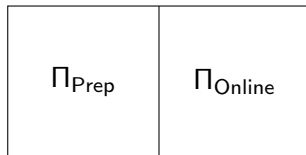
(etc.)

Pre-processing Model



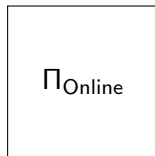
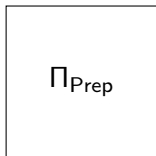
- ▶ Offline phase: computed whenever, uses public-key crypto
- ▶ Online phase: only uses information-theoretic primitives

Pre-processing Model



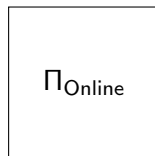
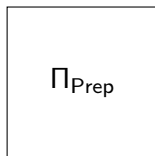
- ▶ Offline phase: computed whenever, uses public-key crypto
- ▶ Online phase: only uses information-theoretic primitives

Pre-processing Model



- ▶ Offline phase: computed whenever, uses public-key crypto
- ▶ Online phase: only uses information-theoretic primitives

Pre-processing Model



- ▶ Offline phase: computed whenever, uses public-key crypto
- ▶ Online phase: only uses information-theoretic primitives

Goal

Take SPDZ, find simple method to outsource Π_{Prep} to different set of parties.

Outline

MPC

Overview

Pre-processing Model

SPDZ Family of MPC

Linear Secret-sharing

Opening secrets

Multiplication of secrets

Generating Pre-processed Data

Protocol

Linear Secret-sharing

Secret x is shared via $\sum_{i=1}^n x_i = x$, where P_i holds x_i . Want to compute an arithmetic circuit.

Linear Secret-sharing

Secret x is shared via $\sum_{i=1}^n x_i = x$, where P_i holds x_i . Want to compute an arithmetic circuit.

Parties can locally:

Add secrets

x_1, y_1

P_1

P_2

x_2, y_2

P_3

x_3, y_3

Linear Secret-sharing

Secret x is shared via $\sum_{i=1}^n x_i = x$, where P_i holds x_i . Want to compute an arithmetic circuit.

Parties can locally:

Add secrets

$$\hat{x}_1 := x_1 + y_1$$

P_1

$$\sum_{i=1}^3 \hat{x}_i = x + y$$

P_2

$$\hat{x}_2 := x_2 + y_2$$

P_3

$$\hat{x}_3 := x_3 + y_3$$

Linear Secret-sharing

Secret x is shared via $\sum_{i=1}^n x_i = x$, where P_i holds x_i . Want to compute an arithmetic circuit.

Parties can locally:

Multiply a secret by a public constant

k, x_1

P_1

P_2

k, x_2

P_3

k, x_3

Linear Secret-sharing

Secret x is shared via $\sum_{i=1}^n x_i = x$, where P_i holds x_i . Want to compute an arithmetic circuit.

Parties can locally:

Multiply a secret by a public constant

$$\hat{x}_1 := k \cdot x_1$$

P_1

$$\sum_{i=1}^3 \hat{x}_i = k \cdot x$$

P_2

$$\hat{x}_2 := k \cdot x_2$$

P_3

$$\hat{x}_3 := k \cdot x_3$$

Linear Secret-sharing

Secret x is shared via $\sum_{i=1}^n x_i = x$, where P_i holds x_i . Want to compute an arithmetic circuit.

Parties can locally:

Add a public constant to a secret

k, x_1

P_1

P_2

k, x_2

P_3

k, x_3

Linear Secret-sharing

Secret x is shared via $\sum_{i=1}^n x_i = x$, where P_i holds x_i . Want to compute an arithmetic circuit.

Parties can locally:

Add a public constant to a secret

$$\hat{x}_1 := x_1 + k$$

P_1

$$\sum_{i=1}^3 \hat{x}_i = x + k$$

P_2

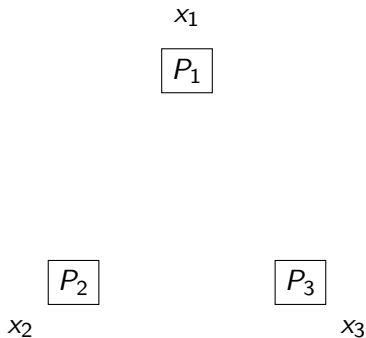
$$\hat{x}_2 = x_2$$

P_3

$$\hat{x}_3 = x_3$$

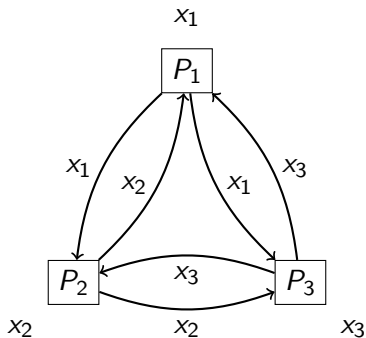
Opening a secret

To reveal or *open* a secret, the parties broadcast their share:



Opening a secret

To reveal or *open* a secret, the parties broadcast their share:



Opening a secret

To reveal or *open* a secret, the parties broadcast their share:

x_1, x_2, x_3

P_1

$$x = x_1 + x_2 + x_3$$

P_2

x_2, x_3, x_1

P_3

x_3, x_1, x_2

Multiplication?

Want $(z_i)_{i=1}^3$ s.t. $\sum_{i=1}^3 z_i = xy$.

Multiplication?

Want $(z_i)_{i=1}^3$ s.t. $\sum_{i=1}^3 z_i = xy$.

Observe that:

$$\begin{aligned}xy &= (x - a + a)(y - b + b) \\ &= (x - a)(y - b) + (x - a)b + a(y - b) + ab\end{aligned}$$

Multiplication?

Want $(z_i)_{i=1}^3$ s.t. $\sum_{i=1}^3 z_i = xy$.

Observe that:

$$\begin{aligned}xy &= (x - a + a)(y - b + b) \\ &= (x - a)(y - b) + (x - a)b + a(y - b) + ab\end{aligned}$$

so if the parties *already* have

shares of two random secrets, $a = \sum_{i=1}^3 a_i$ and $b = \sum_{i=1}^3 b_i$...

...and a share of their product $ab = \sum_{i=1}^3 c_i$,

then define

$$z_1 := (x - a)(y - b) + (x - a)b_1 + a_1(y - b) + c_1$$

$$z_2 := (x - a)b_2 + a_2(y - b) + c_2$$

$$z_3 := (x - a)b_3 + a_3(y - b) + c_3$$

Multiplication?

Key points:

- ▶ Need a triple: shared random independent a and b and sharing of ab .
- ▶ Triple (a, b, c) is *independent* of x and y
- ▶ Requires 2 openings
- ▶ None of the secrets x , y , or xy are revealed

Outline

MPC

- Overview

- Pre-processing Model

SPDZ Family of MPC

- Linear Secret-sharing

- Opening secrets

- Multiplication of secrets

Generating Pre-processed Data

Protocol

Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE

$$a_1, b_1 \leftarrow \mathbb{F}$$

$$\boxed{P_1}$$

$$\boxed{P_2}$$

$$a_2, b_2 \leftarrow \mathbb{F}$$

$$\boxed{P_3}$$

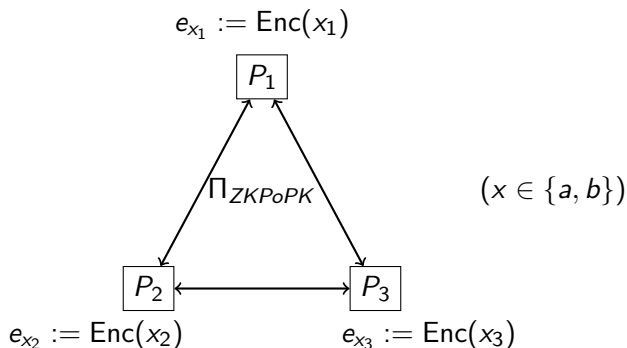
$$a_3, b_3 \leftarrow \mathbb{F}$$

Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs)

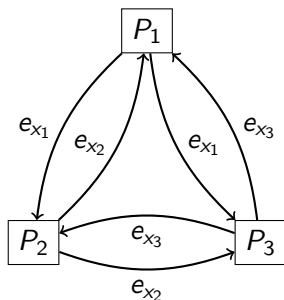


Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs)



$(x \in \{a, b\})$

Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs)

$$e_x := e_{x_1} + e_{x_2} + e_{x_3}$$

$$\boxed{P_1}$$

$$(x \in \{a, b\})$$

$$\boxed{P_2}$$

$$e_x := e_{x_1} + e_{x_2} + e_{x_3}$$

$$\boxed{P_3}$$

$$e_x := e_{x_1} + e_{x_2} + e_{x_3}$$

Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs)

$$e_c := e_a \times e_b$$

$$P_1$$

$$P_2$$

$$e_c := e_a \times e_b$$

$$P_3$$

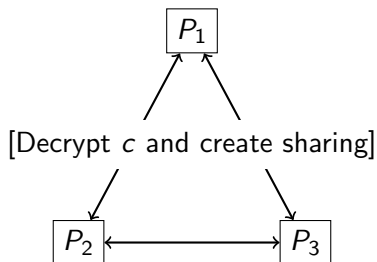
$$e_c := e_a \times e_b$$

Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs)



Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs)

a_1, b_1, c_1

P_1

P_2

a_2, b_2, c_2

P_3

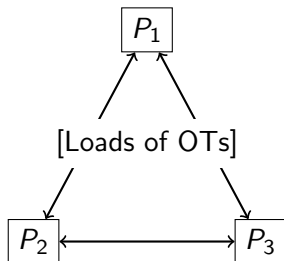
a_3, b_3, c_3

Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs), or
- ▶ OT



Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs), or
- ▶ OT

a_1, b_1, c_1

P_1

P_2

a_2, b_2, c_2

P_3

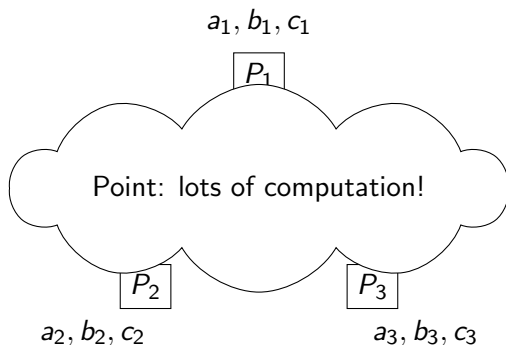
a_3, b_3, c_3

Generating Pre-processing in Π_{Prep}

Need to generate Beaver Triples.

Generated via:

- ▶ SHE (+ ZKPs), or
- ▶ OT



Idea

Problem: in pre-processing,

- ▶ lots of random data is required;
- ▶ public-key crypto is used a lot.

Idea

Problem: in pre-processing,

- ▶ lots of random data is required;
- ▶ public-key crypto is used a lot.

Goal

Take SPDZ, find simple method to outsource Π_{Prep} to different set of parties.

Idea

Problem: in pre-processing,

- ▶ lots of random data is required;
- ▶ public-key crypto is used a lot.

Goal

Take SPDZ, find simple method to outsource Π_{Prep} to different set of parties.

Solution:

- ▶ Outsource to some set of parties R .
- ▶ Make R do all pre-processing.
- ▶ When computing parties Q need pre-processed data, request from parties in R .

Outline

MPC

- Overview

- Pre-processing Model

SPDZ Family of MPC

- Linear Secret-sharing

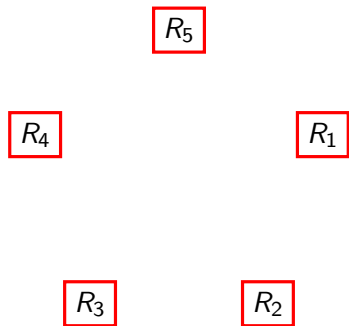
- Opening secrets

- Multiplication of secrets

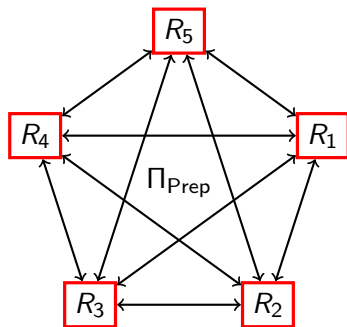
Generating Pre-processed Data

Protocol

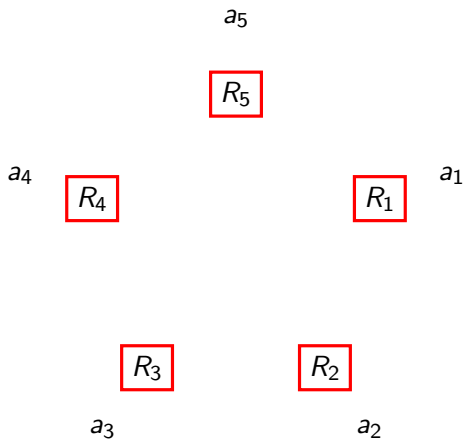
1. Pre-processing



1. Pre-processing



1. Pre-processing



2. Feeding

Π_{Feed}

R_1

R_2

R_3

R_4

R_5

Q_1

Q_2

Q_3

Q_4

Q_5

Q_6

Q_7

2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2$$

R_1

R_2

R_3

R_4

R_5

Q_1

Q_2

Q_3

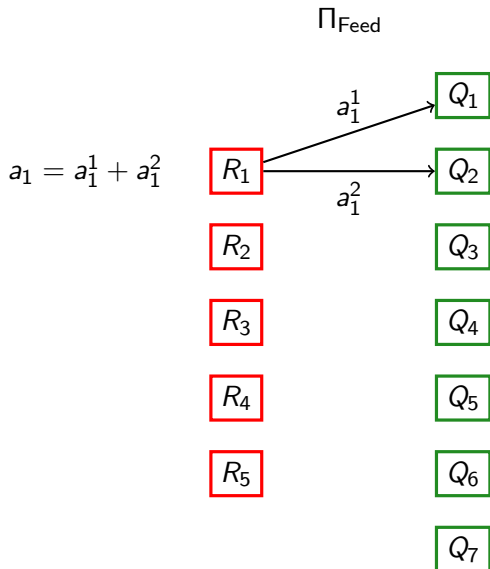
Q_4

Q_5

Q_6

Q_7

2. Feeding



2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2$$

R_1

R_2

R_3

R_4

R_5

Q_1 a_1^1

Q_2 a_1^2

Q_3

Q_4

Q_5

Q_6

Q_7

2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2$$

R_1

$$a_2 = a_2^4 + a_2^7$$

R_2

R_3

R_4

R_5

Q_1 a_1^1

Q_2 a_1^2

Q_3

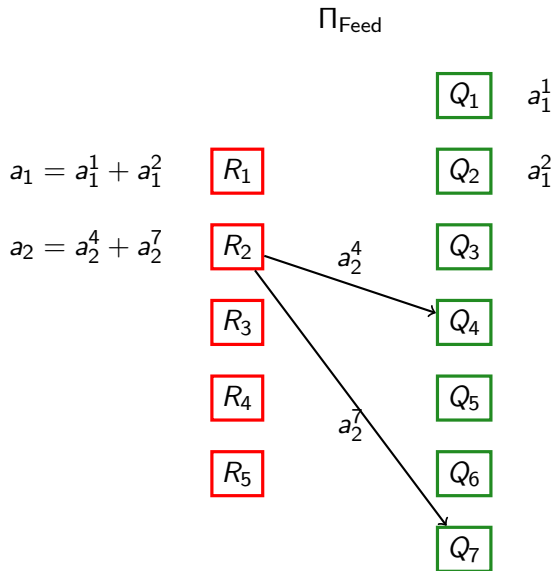
Q_4

Q_5

Q_6

Q_7

2. Feeding



2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2$$

R_1

$$a_2 = a_2^4 + a_2^7$$

R_2

R_3

R_4

R_5

Q_1 a_1^1

Q_2 a_1^2

Q_3

Q_4 a_2^4

Q_5

Q_6

Q_7 a_2^7

2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2$$

R_1

$$a_2 = a_2^4 + a_2^7$$

R_2

$$a_3 = a_3^2 + a_3^3$$

R_3

R_4

R_5

Q_1 a_1^1

Q_2 a_1^2

Q_3

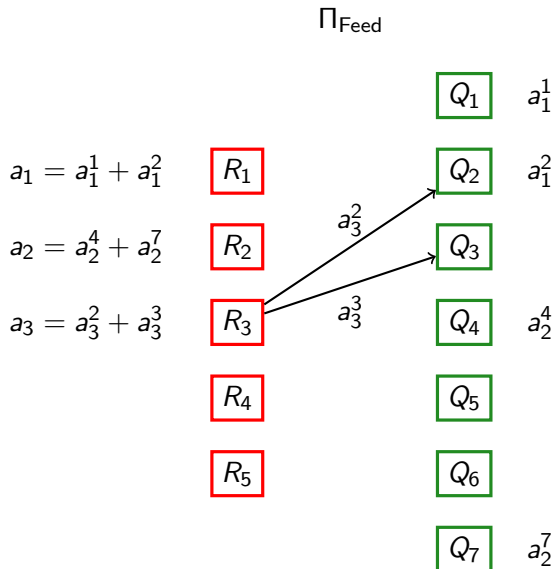
Q_4 a_2^4

Q_5

Q_6

Q_7 a_2^7

2. Feeding



2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2$$

R_1

$$a_2 = a_2^4 + a_2^7$$

R_2

$$a_3 = a_3^2 + a_3^3$$

R_3

R_4

R_5

Q_1 a_1^1

Q_2 a_1^2, a_3^2

Q_3 a_3^3

Q_4 a_2^4

Q_5

Q_6

Q_7 a_2^7

2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2$$

R_1

$$a_2 = a_2^4 + a_2^7$$

R_2

$$a_3 = a_3^2 + a_3^3$$

R_3

$$a_4 = a_4^5 + a_4^7$$

R_4

R_5

Q_1 a_1^1

Q_2 a_1^2, a_3^2

Q_3 a_3^3

Q_4 a_2^4

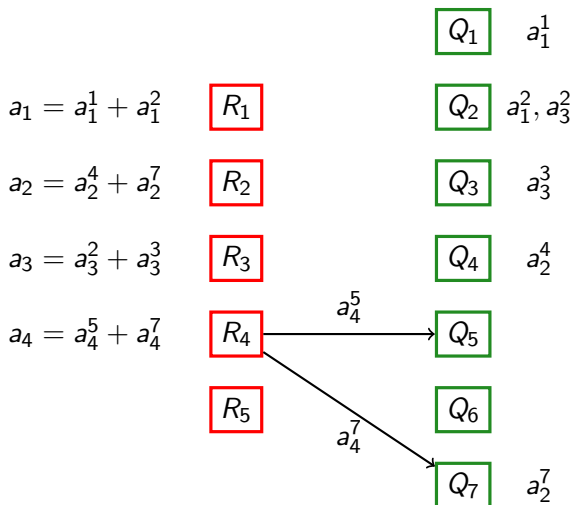
Q_5

Q_6

Q_7 a_2^7

2. Feeding

Π_{Feed}



2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2$$

R_1

$$a_2 = a_2^4 + a_2^7$$

R_2

$$a_3 = a_3^2 + a_3^3$$

R_3

$$a_4 = a_4^5 + a_4^7$$

R_4

R_5

Q_1 a_1^1

Q_2 a_1^2, a_3^2

Q_3 a_3^3

Q_4 a_2^4

Q_5 a_4^5

Q_6

Q_7 a_2^7, a_4^7

2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2 \quad \boxed{R_1}$$

$$a_2 = a_2^4 + a_2^7 \quad \boxed{R_2}$$

$$a_3 = a_3^2 + a_3^3 \quad \boxed{R_3}$$

$$a_4 = a_4^5 + a_4^7 \quad \boxed{R_4}$$

$$a_5 = a_5^1 + a_5^6 \quad \boxed{R_5}$$

$$\boxed{Q_1} \quad a_1^1$$

$$\boxed{Q_2} \quad a_1^2, a_3^2$$

$$\boxed{Q_3} \quad a_3^3$$

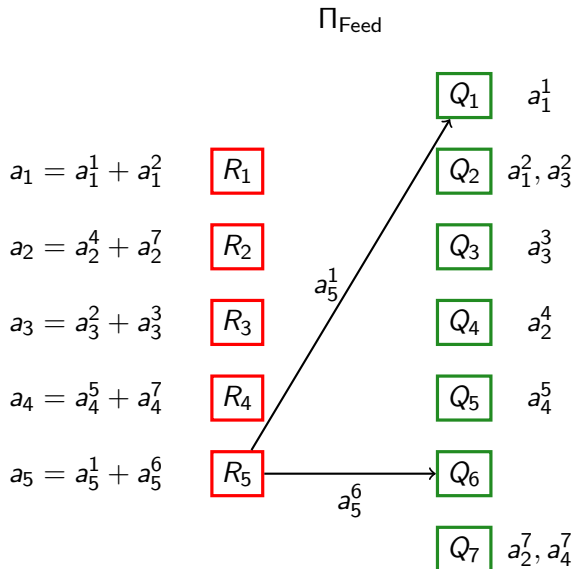
$$\boxed{Q_4} \quad a_2^4$$

$$\boxed{Q_5} \quad a_4^5$$

$$\boxed{Q_6}$$

$$\boxed{Q_7} \quad a_2^7, a_4^7$$

2. Feeding



2. Feeding

Π_{Feed}

$$a_1 = a_1^1 + a_1^2 \quad \boxed{R_1}$$

$$a_2 = a_2^4 + a_2^7 \quad \boxed{R_2}$$

$$a_3 = a_3^2 + a_3^3 \quad \boxed{R_3}$$

$$a_4 = a_4^5 + a_4^7 \quad \boxed{R_4}$$

$$a_5 = a_5^1 + a_5^6 \quad \boxed{R_5}$$

$$\boxed{Q_1} \quad a_1^1, a_5^1$$

$$\boxed{Q_2} \quad a_1^2, a_3^2$$

$$\boxed{Q_3} \quad a_3^3$$

$$\boxed{Q_4} \quad a_2^4$$

$$\boxed{Q_5} \quad a_4^5$$

$$\boxed{Q_6} \quad a_5^6$$

$$\boxed{Q_7} \quad a_2^7, a_4^7$$

2. Feeding

$$\sum_{i=1}^5 a_i = \sum_{j=1}^7 a^j$$

Π_{Feed}

$$a_1 = a_1^1 + a_1^2 \quad R_1$$

$$a_2 = a_2^4 + a_2^7 \quad R_2$$

$$a_3 = a_3^2 + a_3^3 \quad R_3$$

$$a_4 = a_4^5 + a_4^7 \quad R_4$$

$$a_5 = a_5^1 + a_5^6 \quad R_5$$

$$Q_1 \quad a_1^1, a_5^1 \quad a^1 := a_1^1 + a_5^1$$

$$Q_2 \quad a_1^2, a_3^2 \quad a^2 := a_1^2 + a_3^2$$

$$Q_3 \quad a_3^3 \quad a^3 := a_3^3$$

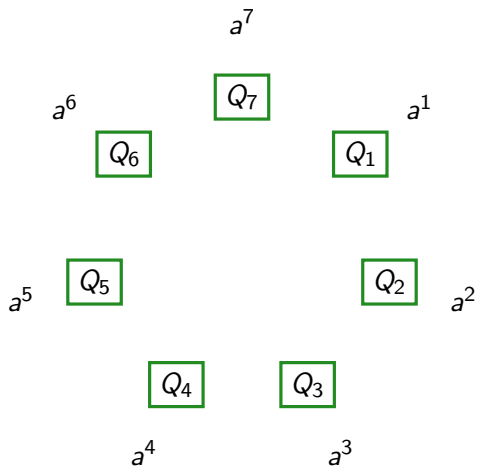
$$Q_4 \quad a_2^4 \quad a^4 := a_2^4$$

$$Q_5 \quad a_4^5 \quad a^5 := a_4^5$$

$$Q_6 \quad a_5^6 \quad a^6 := a_5^6$$

$$Q_7 \quad a_2^7, a_4^7 \quad a^7 := a_2^7 + a_4^7$$

3. Online



Correctness and Privacy

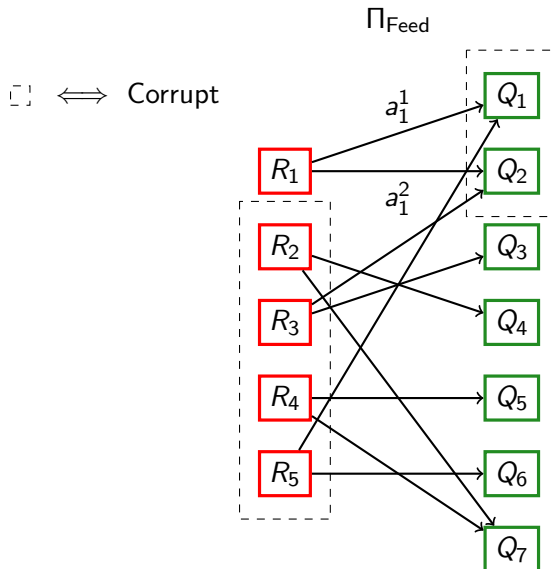
Correctness

Must ensure all parties receive at least one share.

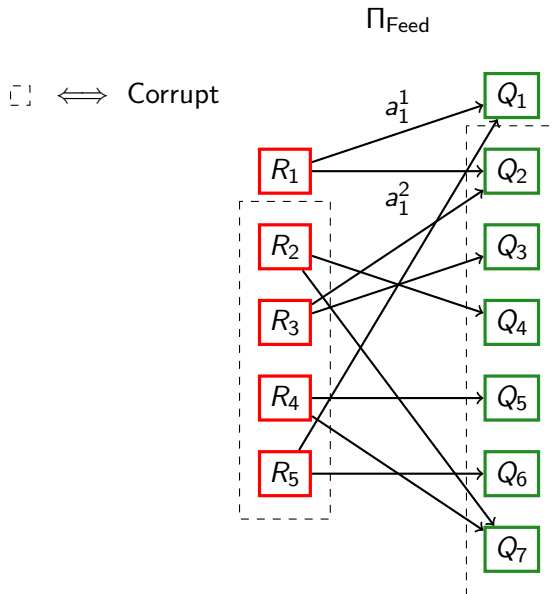
Privacy

?

Q: When does \mathcal{A} not learn the secret?



A: Secure Cover



Ensuring a secure cover

Guaranteed...

...statistically?

- ▶ if cover assigned probabilistically; possible only for large numbers of parties

...perfectly?

- ▶ if $R \subset Q$ (but then a different access structure – why use SPDZ?); or
- ▶ if each party in R reshares to *all* parties in Q

Active Security?

SPDZ uses linear MACS:

$$\text{MAC}_\alpha(a) = \alpha \cdot a$$

Theorem

If adversary corrupts at most all but one party in each of R and Q , and the cover is secure, then the MAC can be forged with probability at most $1/|\mathbb{F}|$.

N.B. in SPDZ, $|\mathbb{F}|$ is $O(2^\lambda)$.

Potential use-cases

- ▶ Client-server system of delivering pre-processing
 - ▶ a few servers compute pre-processing and deliver to a set of 'weaker' clients
- ▶ Dynamically adjusting pre-processing during Π_{Online} if other parties want to join a computation
 - ▶ requires randomness!

Thanks!

Questions?