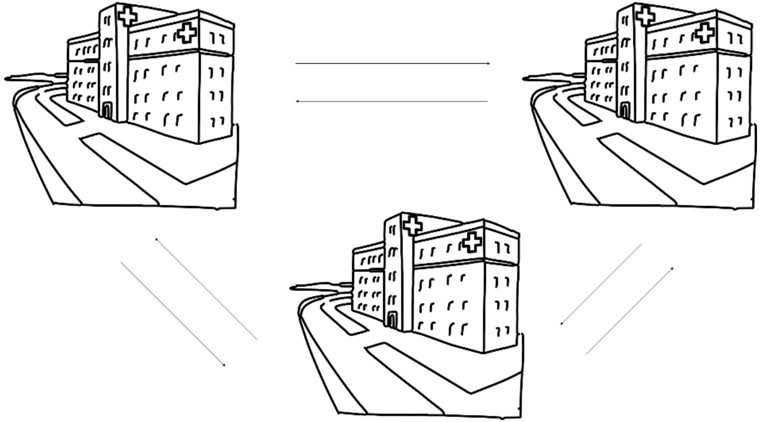
Centrum Wiskunde & Informatica

Welcome!

Secure Computation with Silent Preprocessing



Lisa Kohl, CWI Cryptology Group

Secure Computation

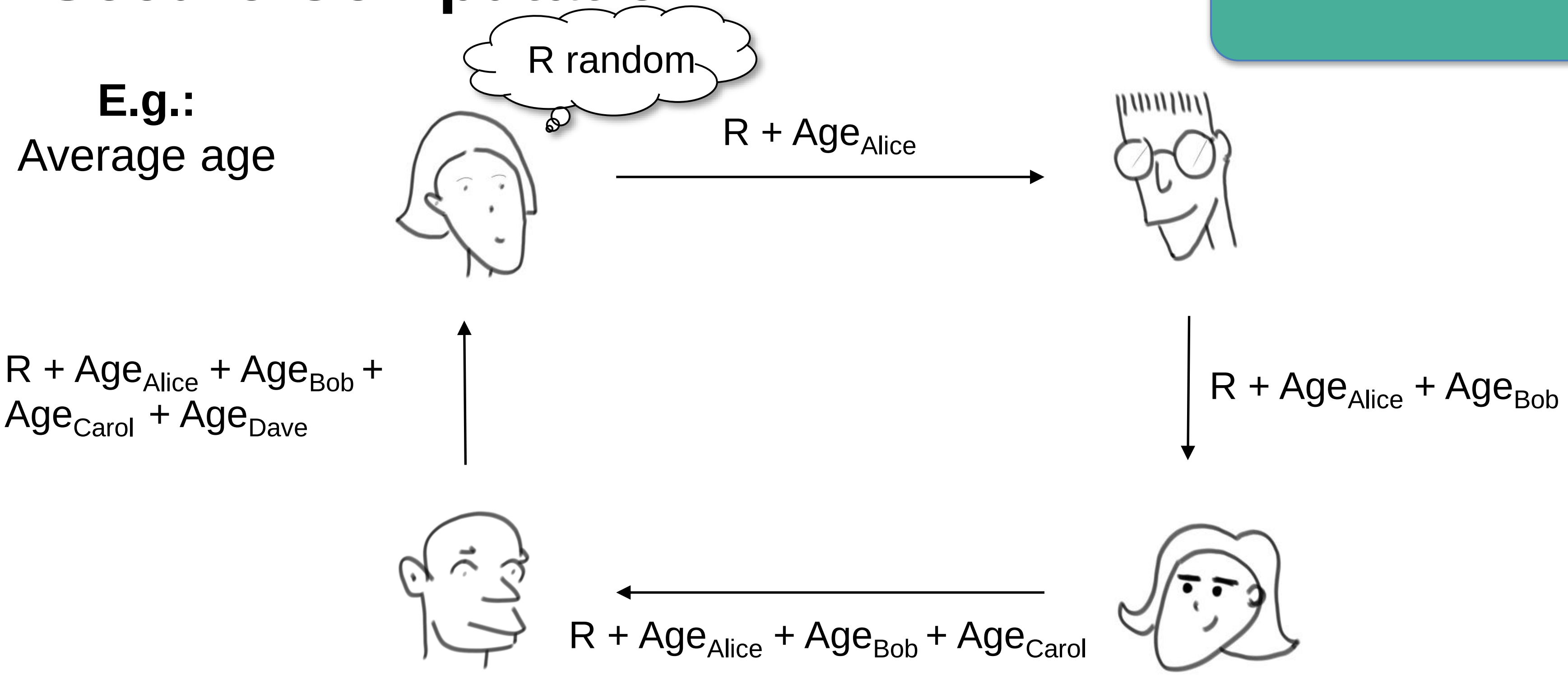
A: Gather the data of all hospitals, to find e.g. optimal treatments.

B: Don‘t gather medical data, because they contain privacy-critical information.

Secure Computation

Secure?

E.g.:
Average age



R random

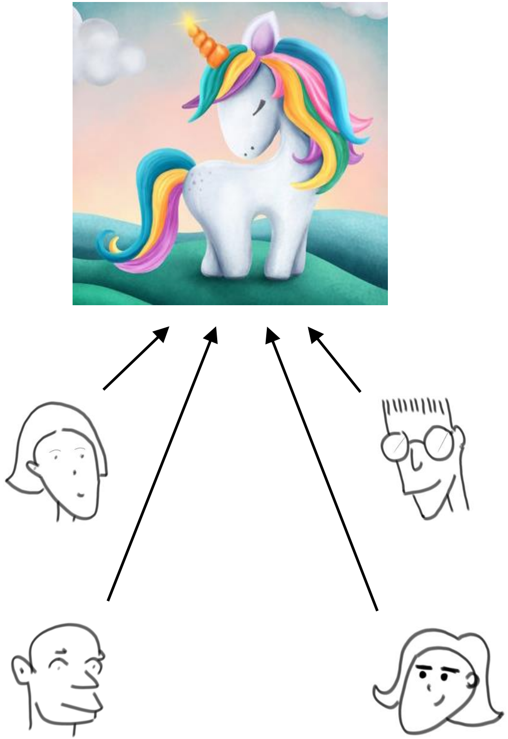
R + $\text{Age}_{\text{Alice}}$

R + $\text{Age}_{\text{Alice}}$ + Age_{Bob}

R + $\text{Age}_{\text{Alice}}$ + Age_{Bob} + $\text{Age}_{\text{Carol}}$

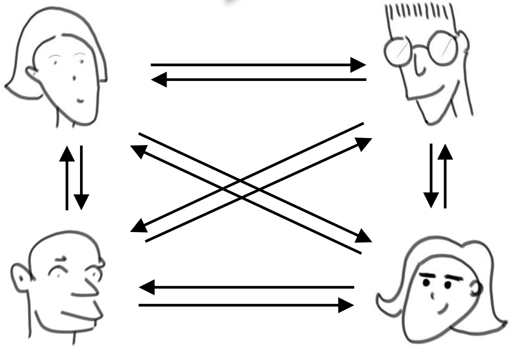
R + $\text{Age}_{\text{Alice}}$ + Age_{Bob} + $\text{Age}_{\text{Carol}}$ + Age_{Dave}

Ideal:



Real:

Secure, if parties cannot learn/ interfere more in the **real world** than in the **ideal world**

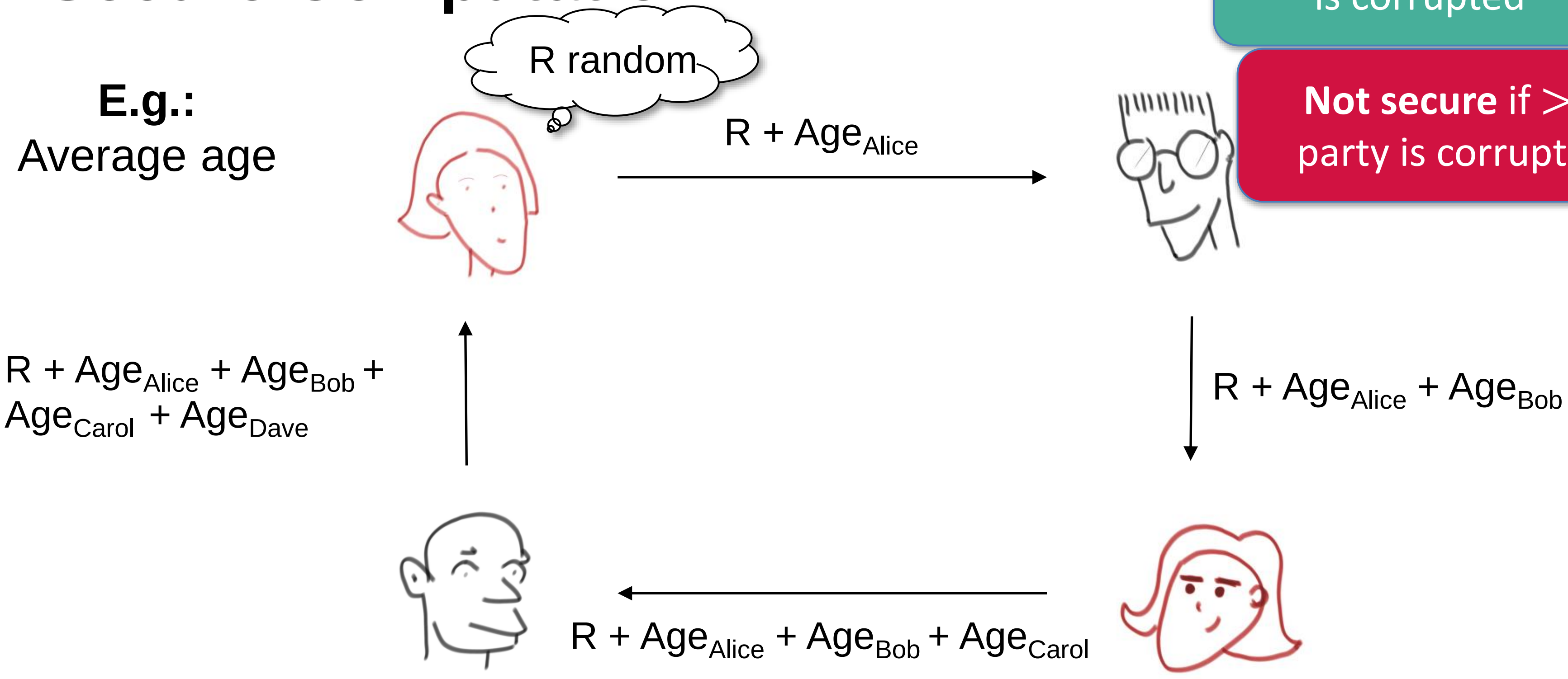


Secure Computation

Secure if ≤ 1 party is corrupted

Not secure if > 1 party is corrupted

E.g.:
Average age



R random

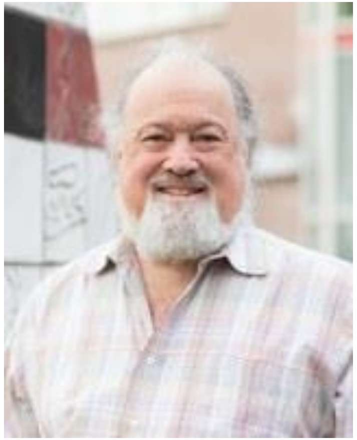
$R + \text{Age}_{\text{Alice}}$

$R + \text{Age}_{\text{Alice}} + \text{Age}_{\text{Bob}}$

$R + \text{Age}_{\text{Alice}} + \text{Age}_{\text{Bob}} + \text{Age}_{\text{Carol}}$

$R + \text{Age}_{\text{Alice}} + \text{Age}_{\text{Bob}} + \text{Age}_{\text{Carol}} + \text{Age}_{\text{Dave}}$

What do we know?



Seminal feasibility results from the 80s:

- **Secure point-to-point channels + broadcast + honest majority:**

  Can compute *any* function (unconditionally)

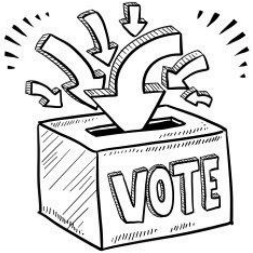
  Ben-Or Goldwasser Wigderson ‘88, Chaum Crépeau Damgård ‘88, Rabin Ben-Or ‘89

- **Public-key cryptography + at least one honest player:**

  Can compute *any* function

  Yao ‘86, Goldreich Micali Wigderson ‘ 87

Why do we care?



- **Electronic auctions, electronic voting**

  E.g.: Financial markets, electricity markets
  Solutions already in use: Partisia (Danish sugar beet auction)

GDPR

- **Privacy-preserving computation on (distributed) databases**

  E.g.: individual HIV treatment, bank fraud detection (TNO/UvA/CWI)
  Solutions already in use: Sharemind

- **Secure set-up of cryptographic infrastructure**

  E.g. Diogenes: secure distributed generation of an RSA modulus

What do we want?

- **Problem:** Generic compilers introduce large overhead

fast algorithm $\not\rightarrow$ **fast secure computation solution**

- **Goal:** Bring secure computation *to every-day life*

- **Possible directions:**
  - Design tailored solutions for specific tasks
  - **Improve generic methods**

My research & this talk

fast algorithm → **fast secure computation solution**

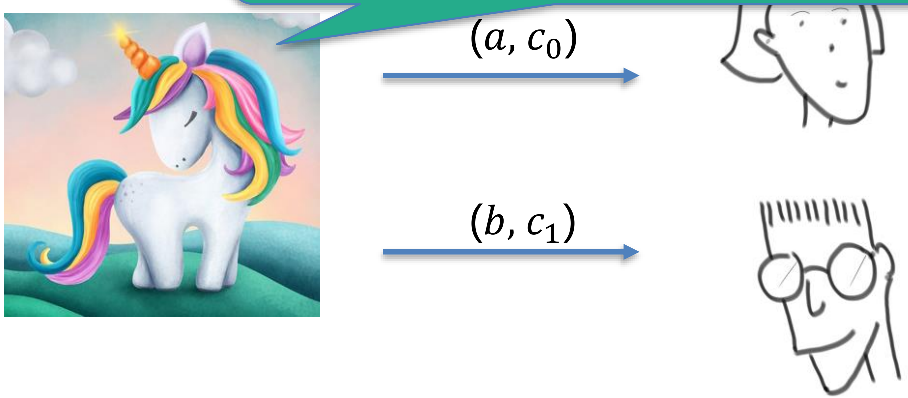
Secure Computation with Preprocessing

Beaver ‘91

- Very efficient online phase
- Security against dishonest majority

a, b, c_0 random,

$$c_0 + c_1 = a \cdot b$$



- **Problem:**
  - Preprocessing communication/ storage ≥ number of multiplications
  - How to generate multiplication tuples securely?

Silent

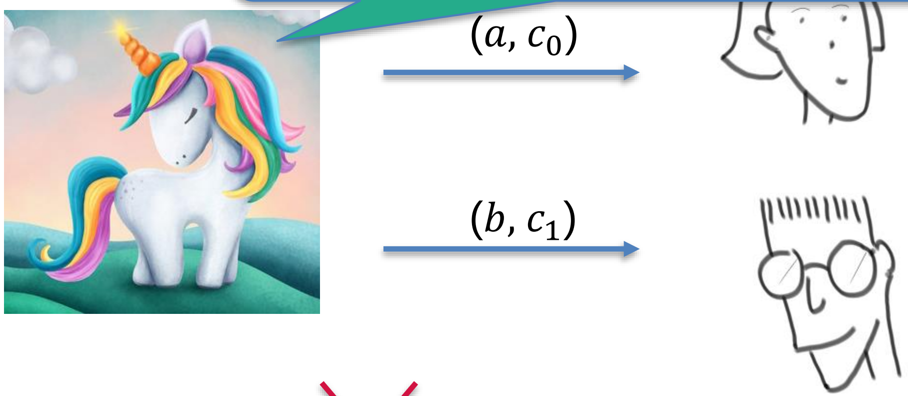
Secure Computation with Preprocessing

Beaver ‘91

- Very efficient online phase
- Security against dishonest majority

a, b, c_0 random,

$$c_0 + c_1 = a \cdot b$$

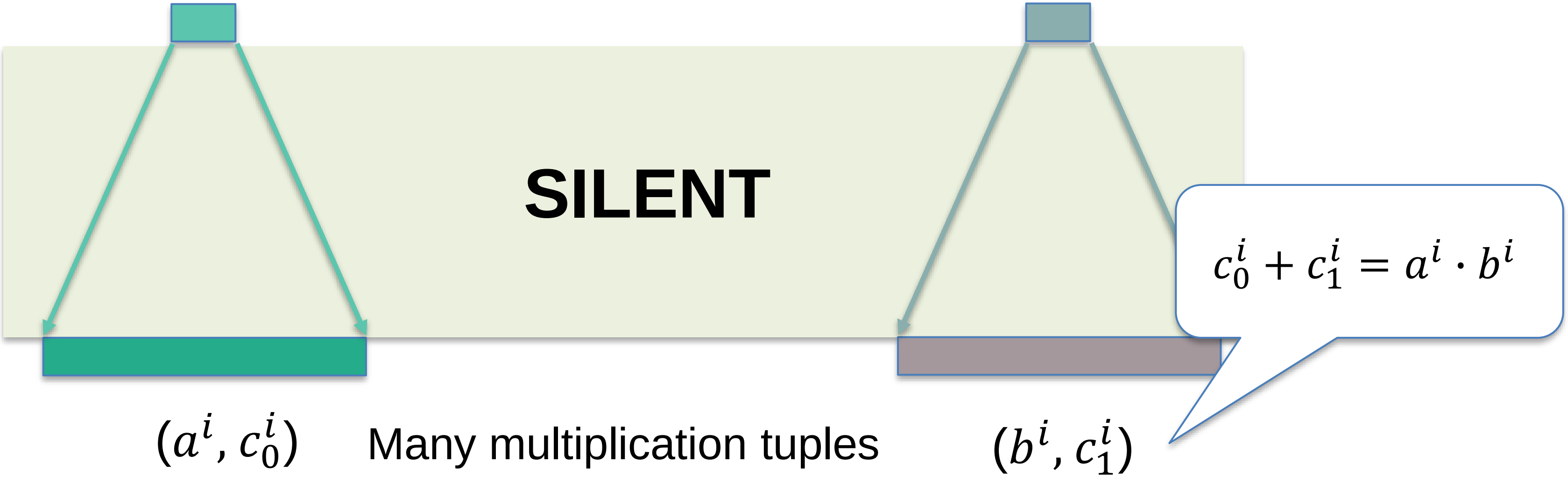


- **Problem:**
  - Preprocessing communication/ storage $\cancel{\geq}$ number of multiplications
  - How to generate multiplication tuples securely?

Secure Computation with Silent Preprocessing

Boyle Couteau Gilboa Ishai **Kohl** Scholl ’19, ‘20a

Goal: Compress multiplication tuples



SILENT

$c_0^i + c_1^i = a^i \cdot b^i$

(a^i, c_0^i) Many multiplication tuples (b^i, c_1^i)

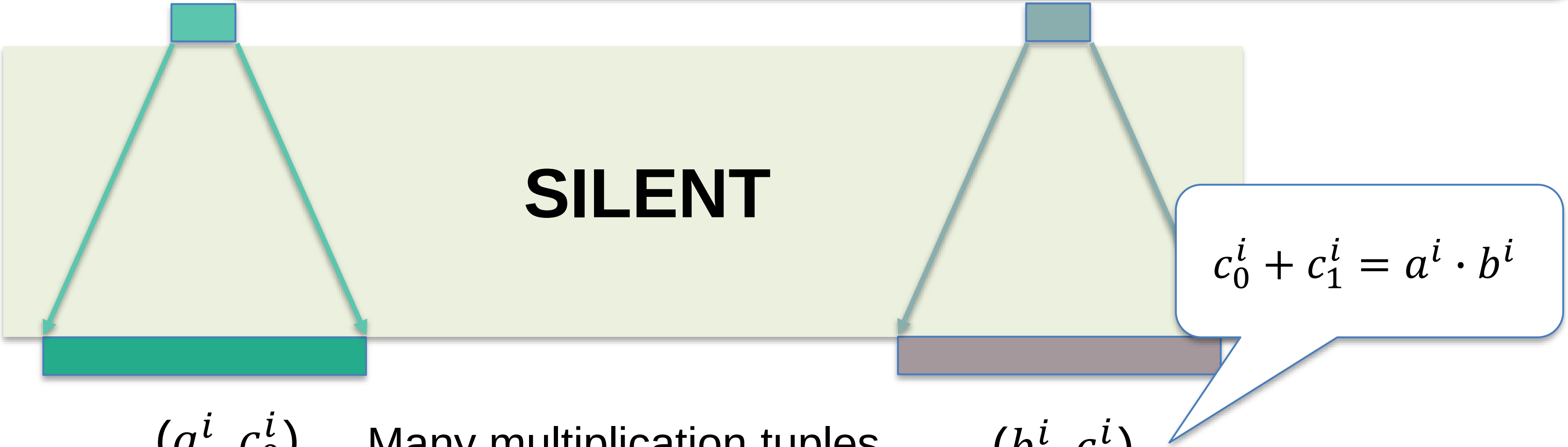
Secure

Can compress (pseudo)randomness via pseudorandom generator

Boyle Coute

Difficulty: Compress c_0^i without revealing $\mathrm{b}^{\mathrm{i}}, c_1^i$ (and vice versa)

Goal: Compre



SILENT

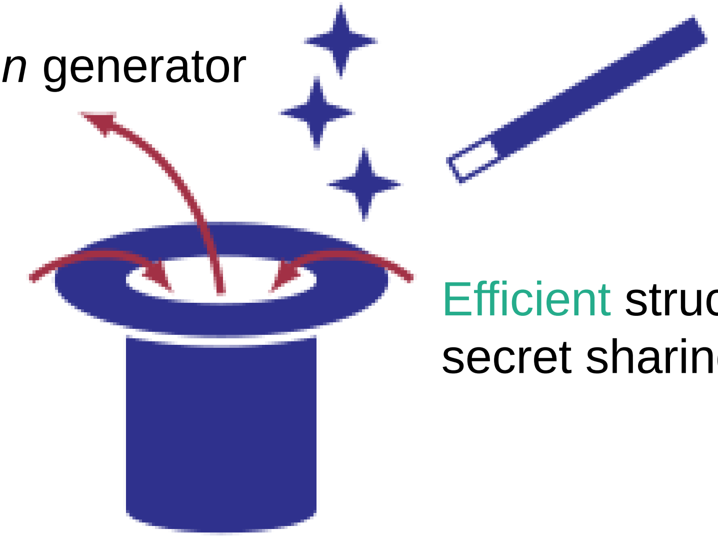
$$c_0^i + c_1^i = a^i \cdot b^i$$

(a^i, c_0^i) Many multiplication tuples (b^i, c_1^i)

Secure Computation with Silent Preprocessing

Boyle Couteau Gilboa Ishai **Kohl** Scholl ’19, ‘20a

Efficient pseudorandom *correlation* generator
e.g. for multiplication tuple



Specifically tailored
pseudorandom generator

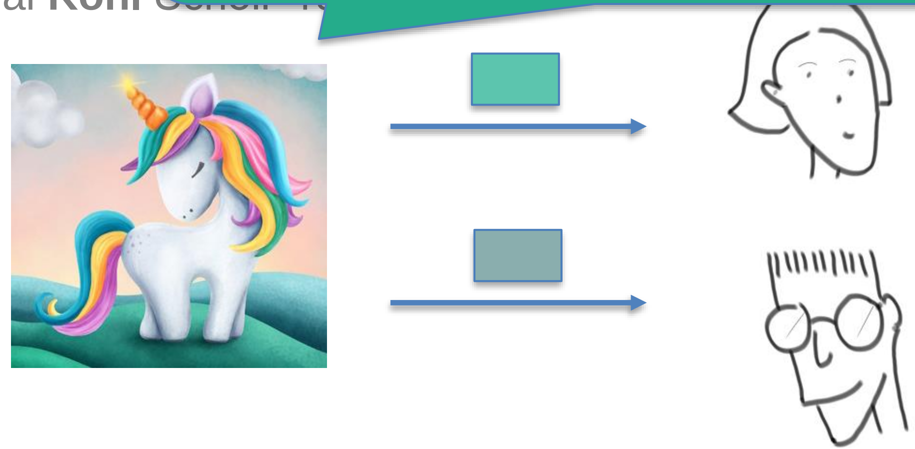
Efficient structure-preserving
secret sharing scheme

Secure Computation

Boyle Couteau Gilboa Ishai **Kohl** S

Concrete efficiency improvements:

- 1000 × less communication
- ≈ 50 × speed-up over slow networks



- Efficient distributed seed generation BCGI**K**S ‘19, ‘20a, BCGI**K**S + Rindal ’19
- Security cannot be broken by known quantum algorithms
- Recent work: Pseudorandom correlation *functions* BCGI**K**S ’20b

Secure Computation

Boyle Couteau Gilboa Ishai **Kohl**

Concrete efficiency improvements:

- 1000 × less communication
- ≈ 50 × speed-up over slow networks

CGIKS + Rindal ’19

S ’20b

Not there yet:

- Most efficient construction & 2-round setup only for Boolean circuits in the 2-party setting
- No efficient constructions at all for some useful correlations

Looking ahead

- **Goal:** Bring secure computation *to every-day life*

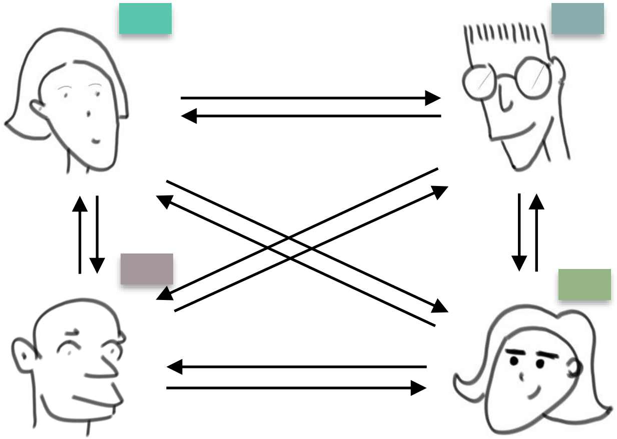
fast algorithm → **fast secure computation solution**

Concrete research goal: ***Efficient fully silent*** **preprocessing in the multi-party setting:**

Everyone publishes short seed *once*

→

Efficient secure computation *for life*



Q&A