MPC-Friendly Symmetric Key Primitives

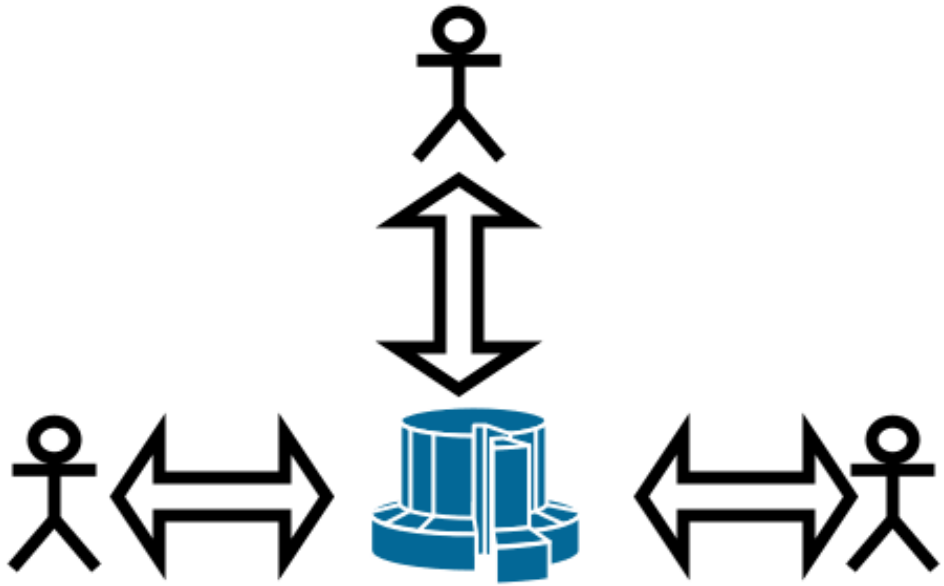
Lorenzo Grassi [1] Christian Rechberger [1] **Dragoș Rotaru** [2]
Peter Scholl [2] Nigel P. Smart [2]

[1]Graz University of Technology

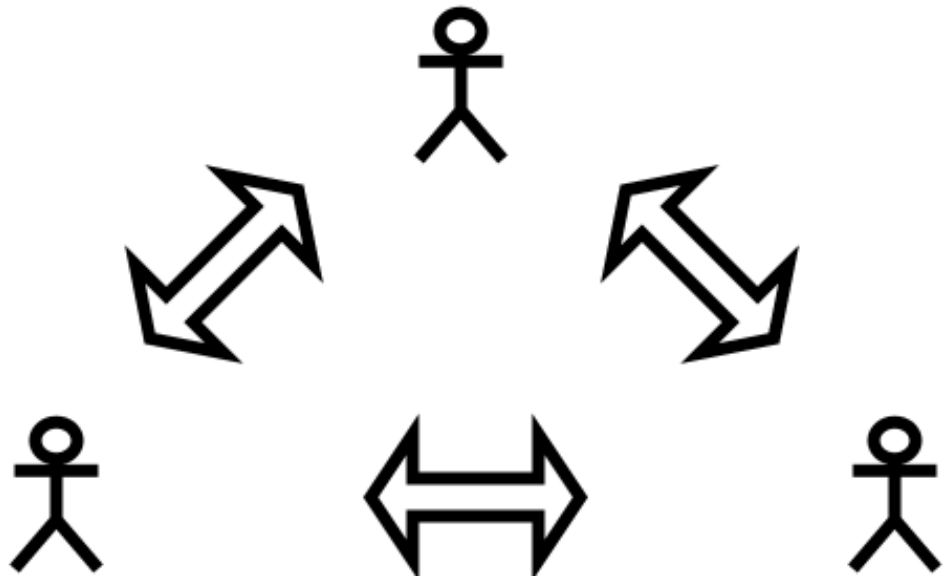
[2]University of Bristol

October 25, 2016

What is Multiparty Computation?



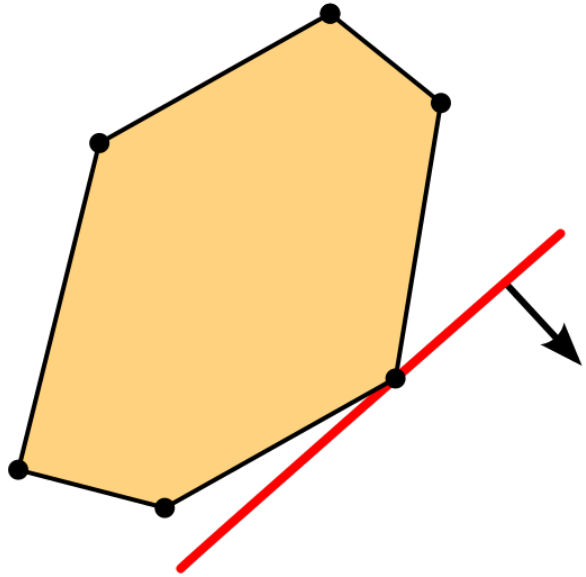
What is Multiparty Computation?



Take home message

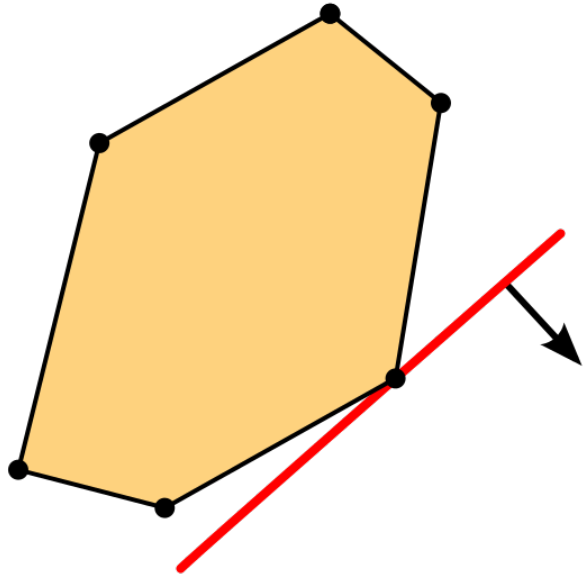
Move data **securely** between clients and MPC engines.

Common to all



Linear Programming

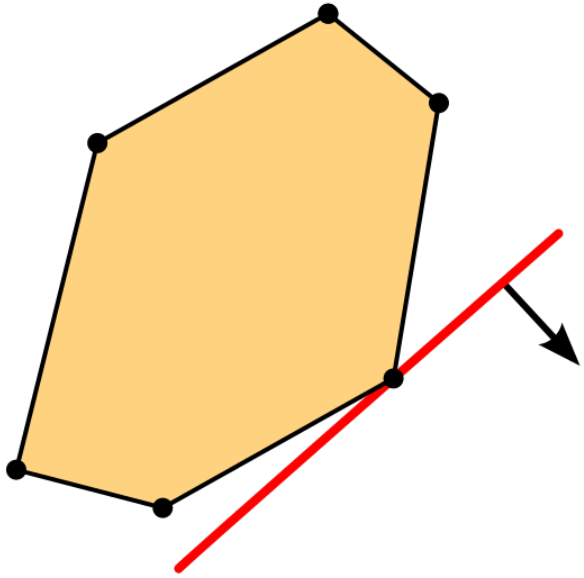
Common to all



Linear Programming

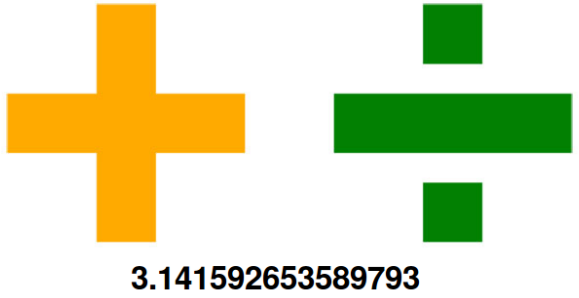
Integer Comparison

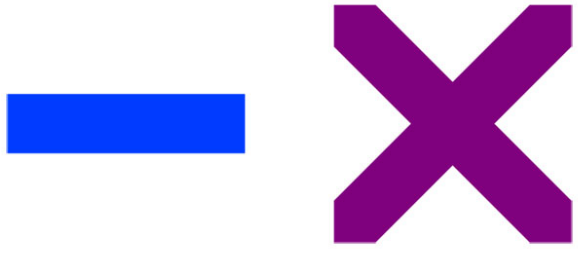
Common to all



Linear Programming

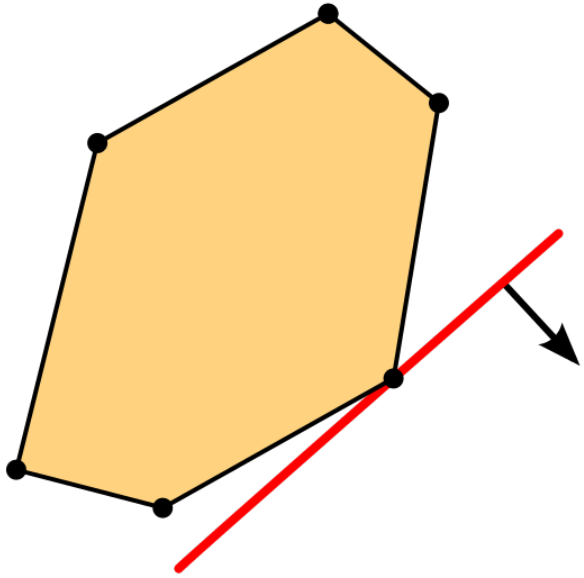
Integer Comparison





Fixed Point Arithmetic

Common to all



Linear Programming

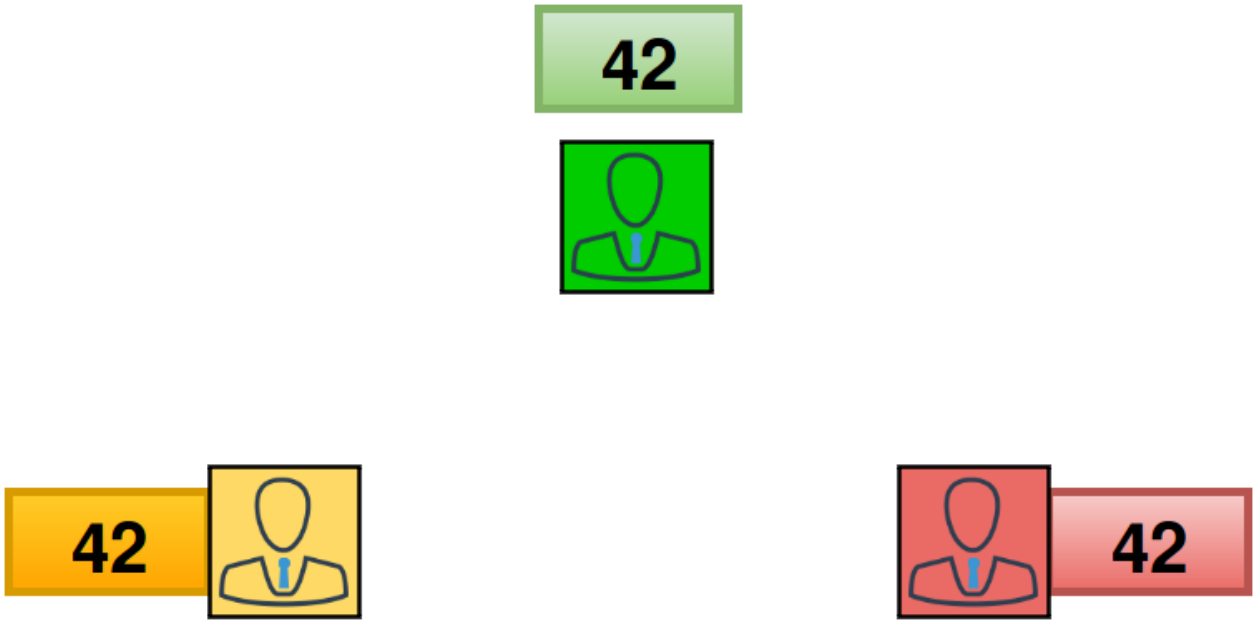
Integer Comparison

3.141592653589793

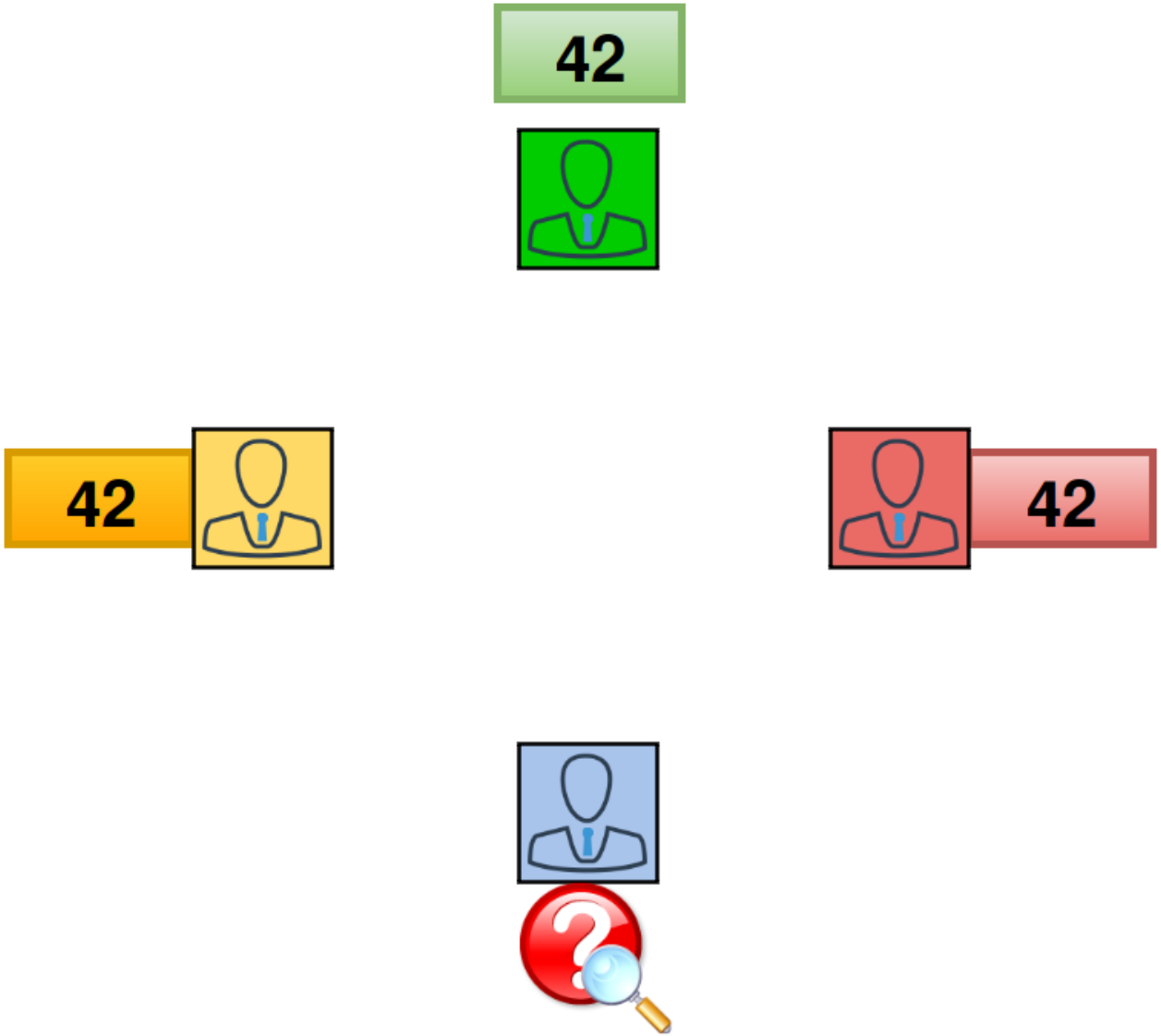
Fixed Point Arithmetic

Easy to implement via arithmetic circuits mod p

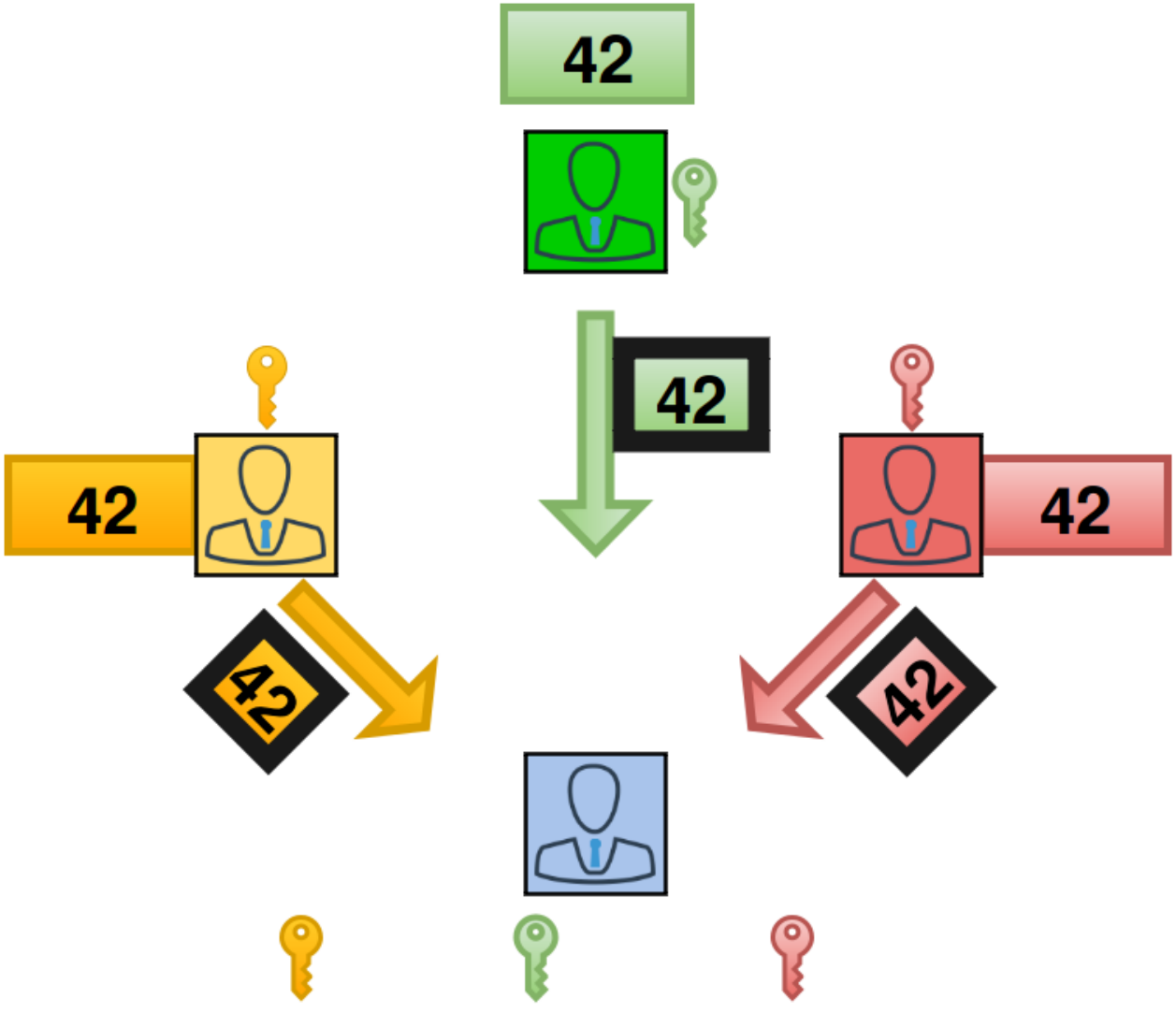
Where is the problem?



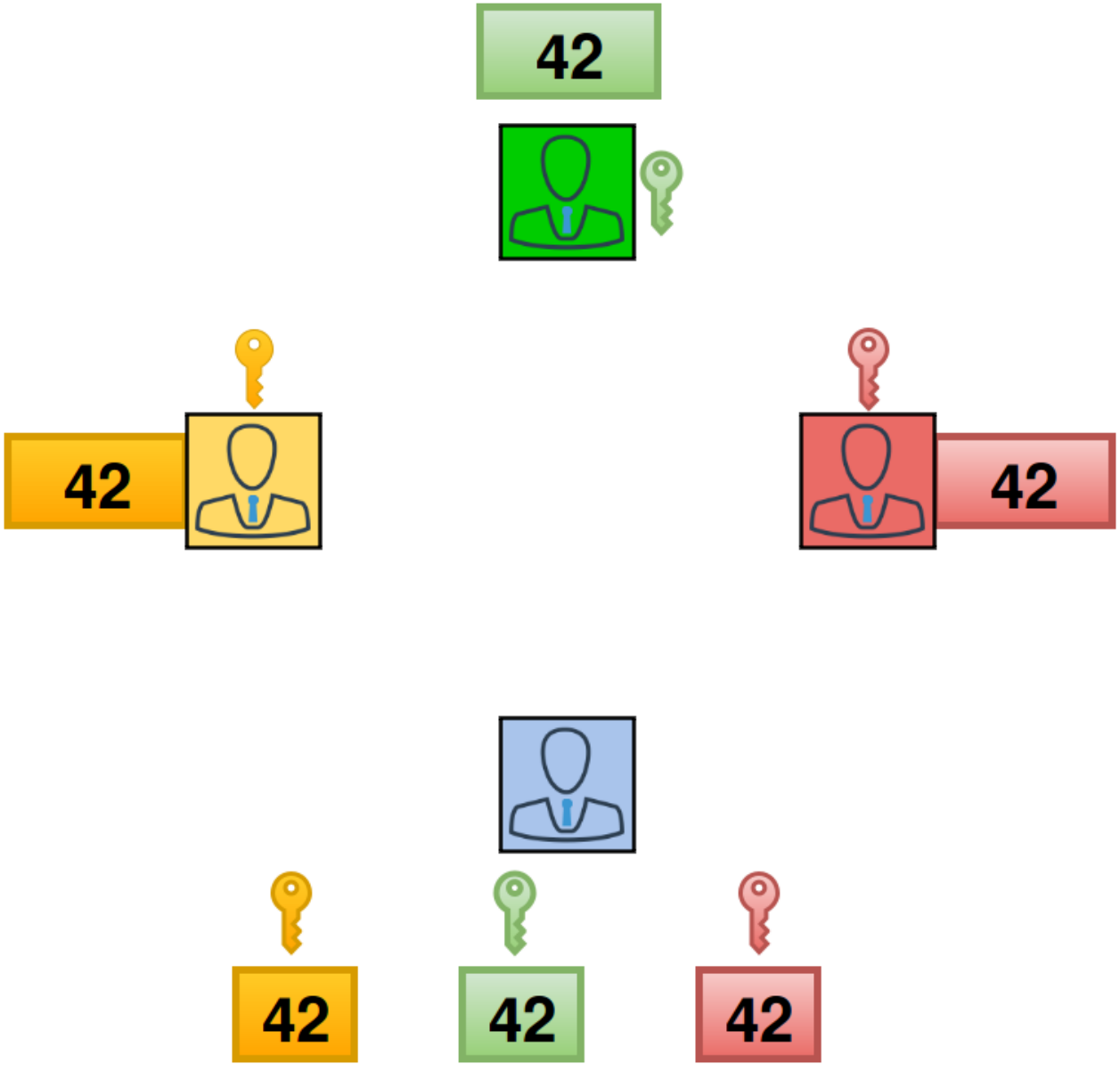
Where is the problem?



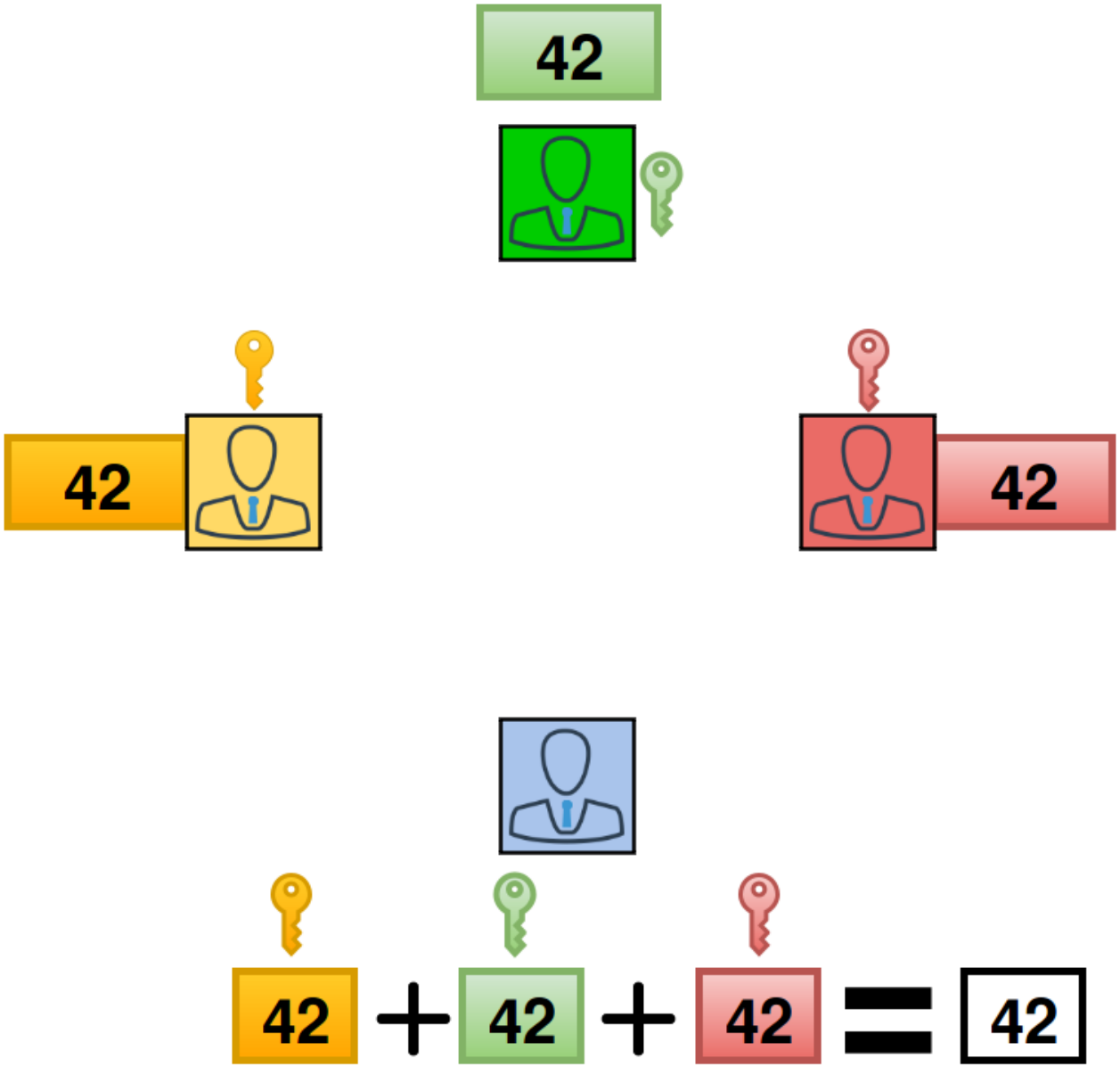
Where is the problem?



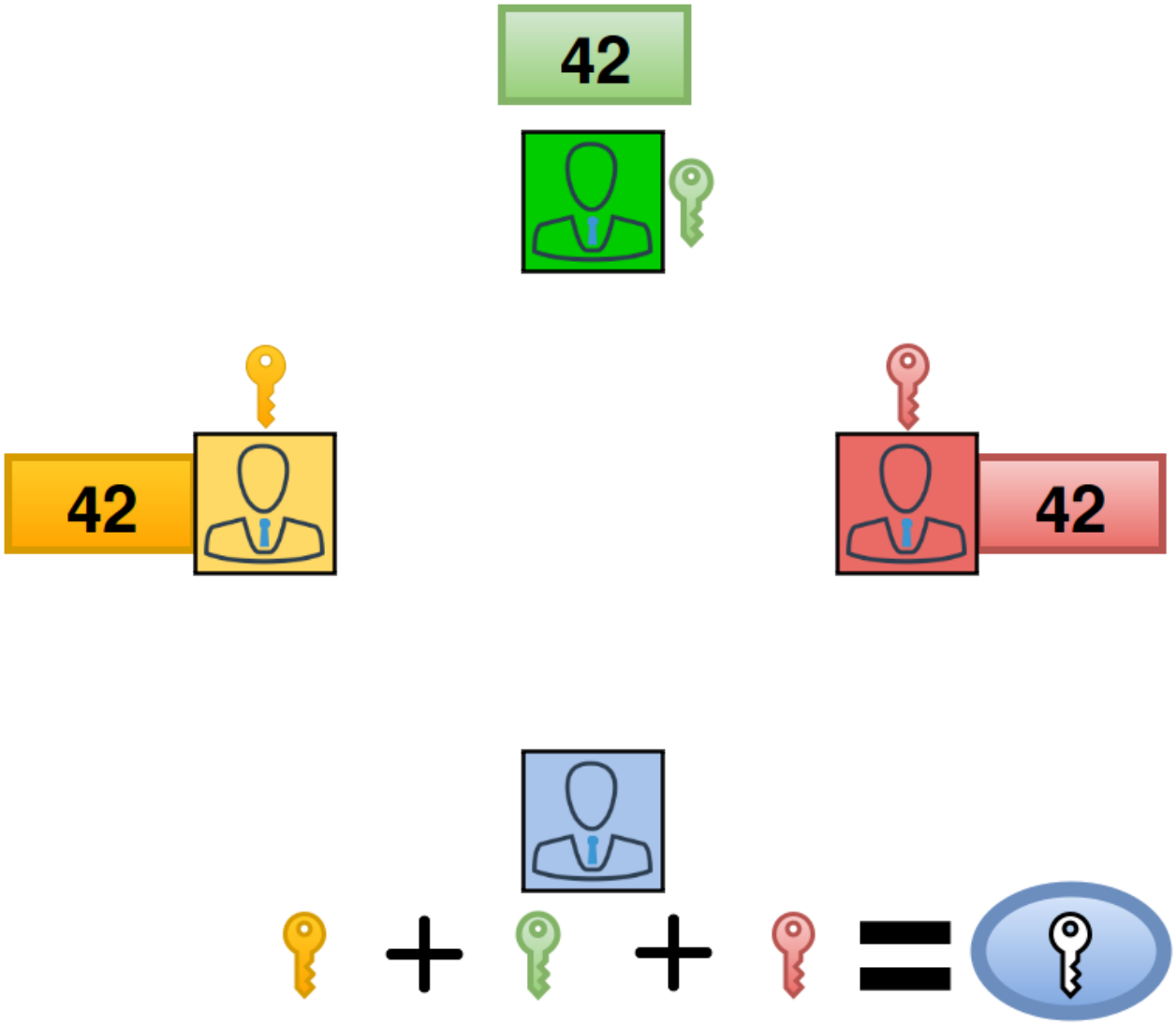
Where is the problem?



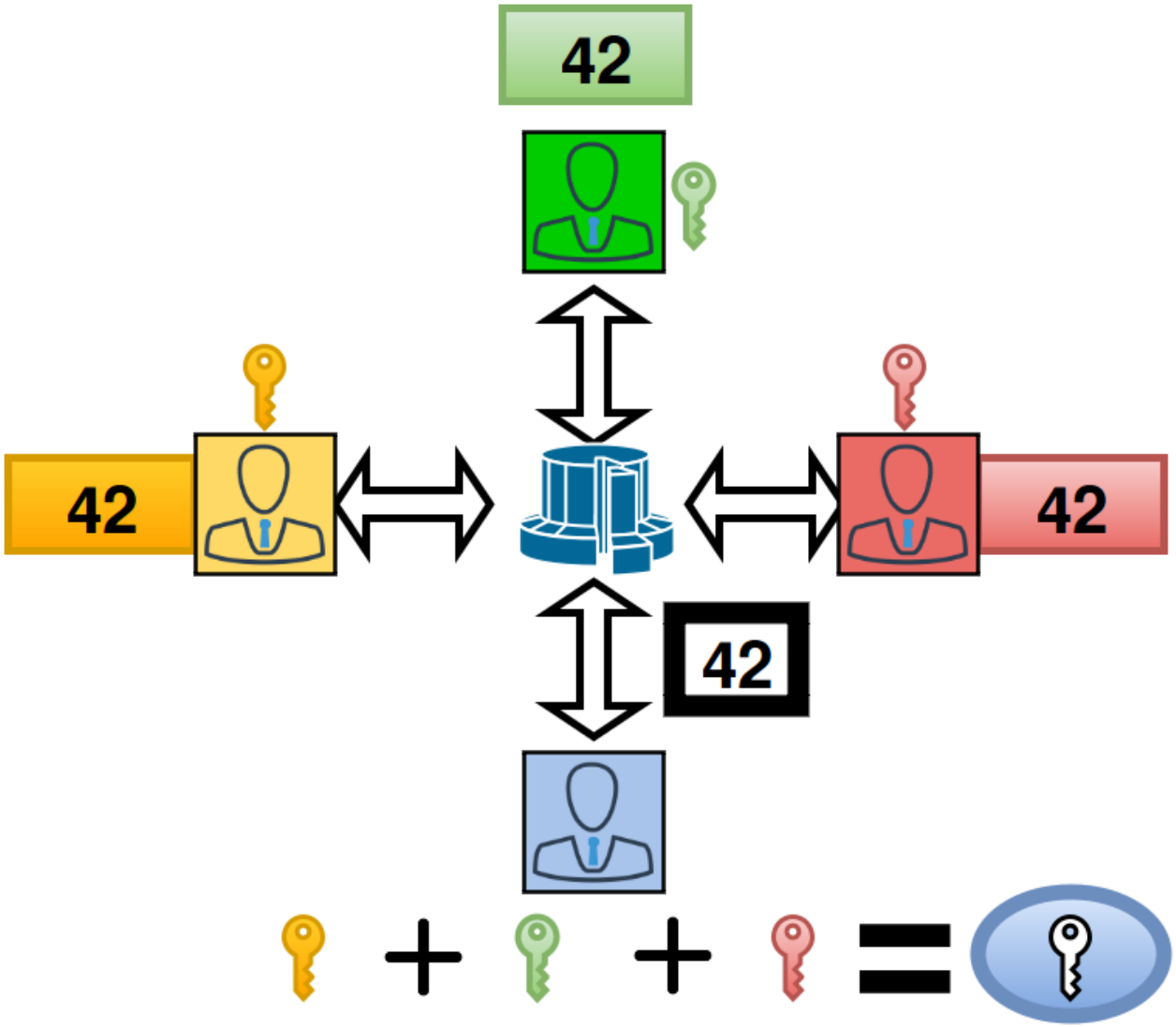
Where is the problem?



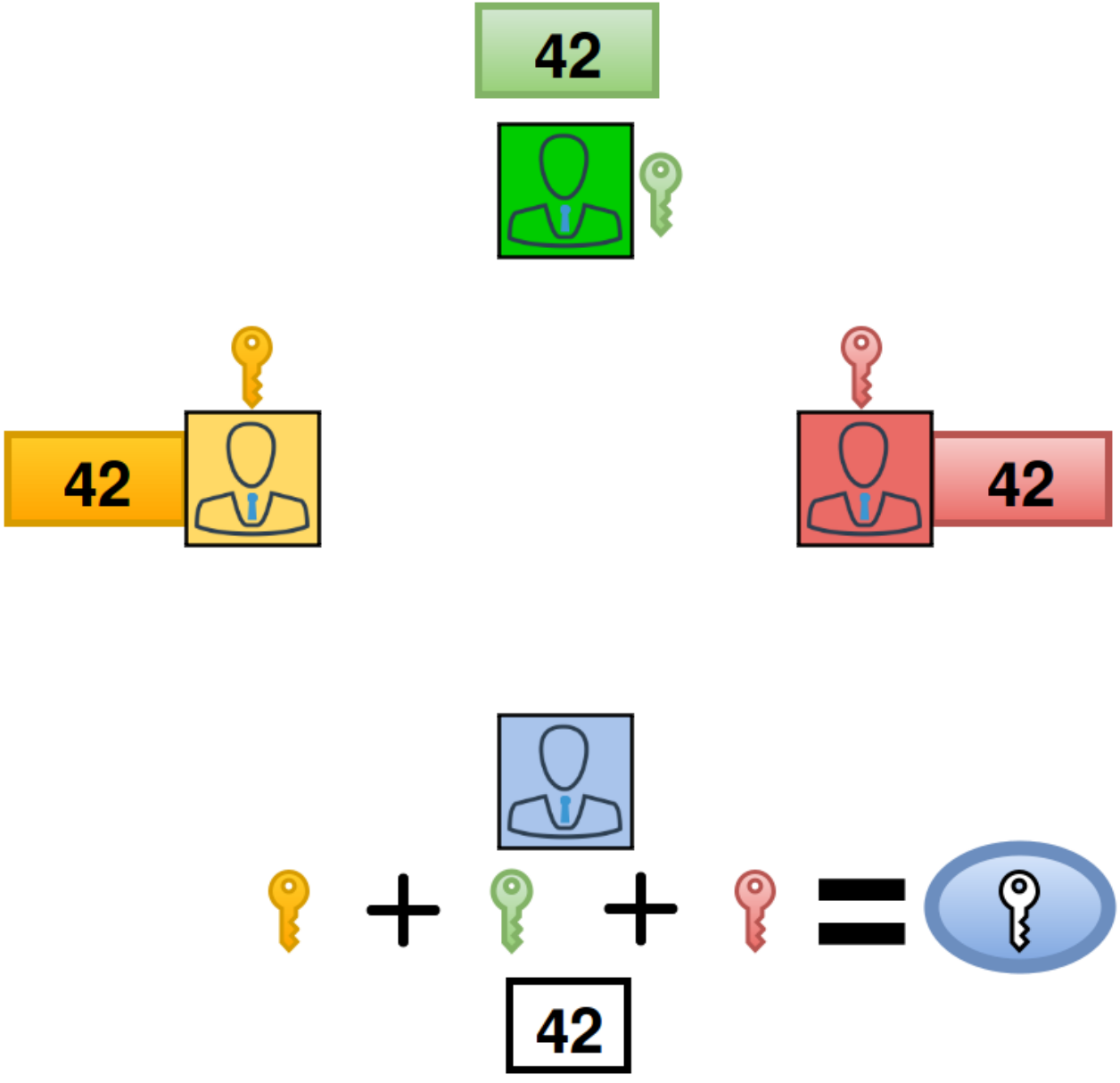
Where is the problem?



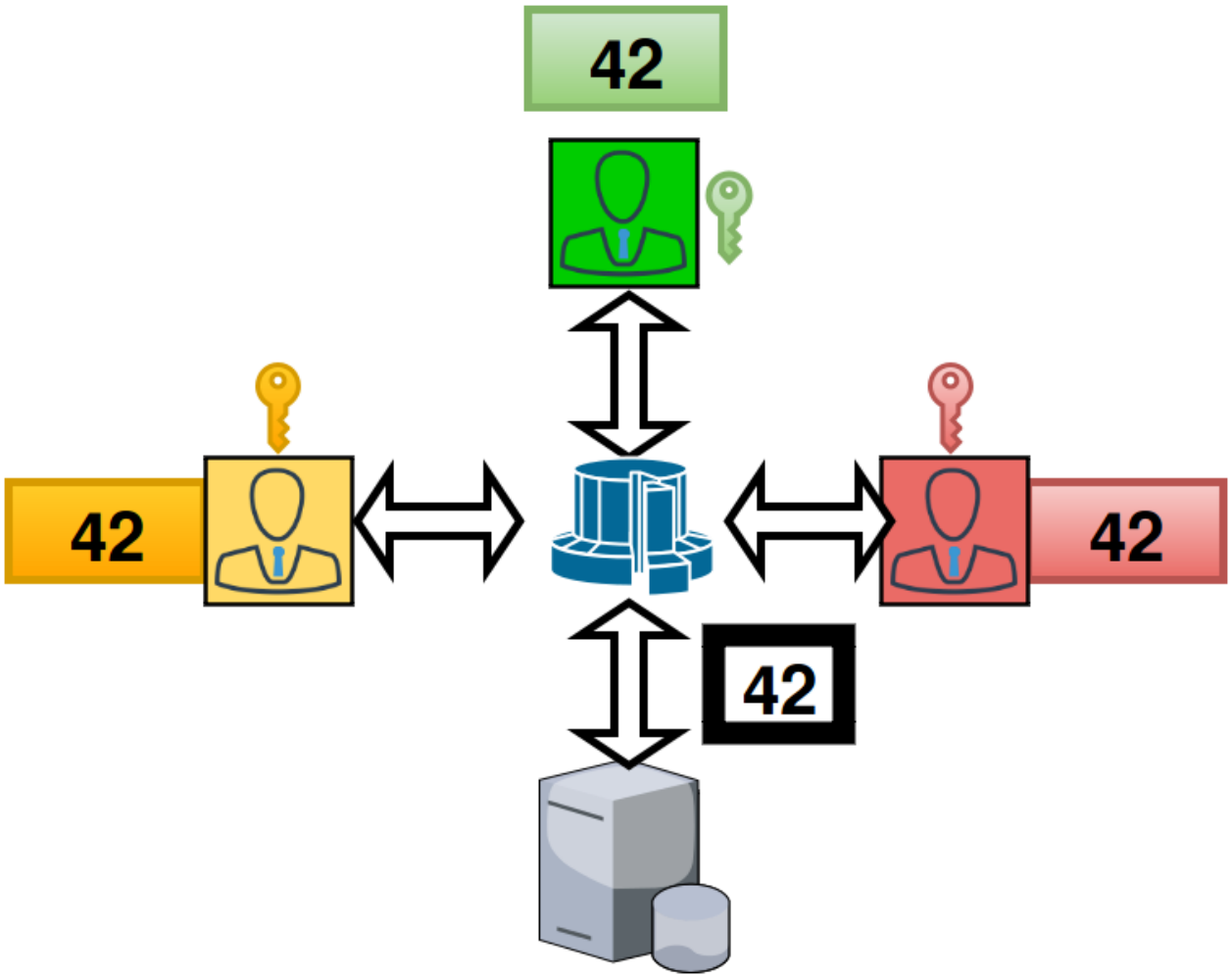
Where is the problem?



Where is the problem?



Where is the problem?



Why?

- Why a PRF? Because we can Enc / Dec using CTR mode (only PRF calls).
- Avoid the n fold database/key blowup by secret share the key and use a PRF mod p in MPC!
- Why mod p? Conversion between binary and arithmetic shares is expensive.

Other use cases for PRF's in MPC

- Secure database joins [LTW13].
- Oblivious RAM [LO13].
- Searchable symmetric encryption, order-revealing encryption [BCO'N11, BLRSZZ15, CLWW16, BBO'N07, CJJKRS13].

What we have done

Benchmark and **create new protocols** using PRF's within SPDZ protocol.

Why SPDZ?

- MPC protocol with active security.
- It is open source.

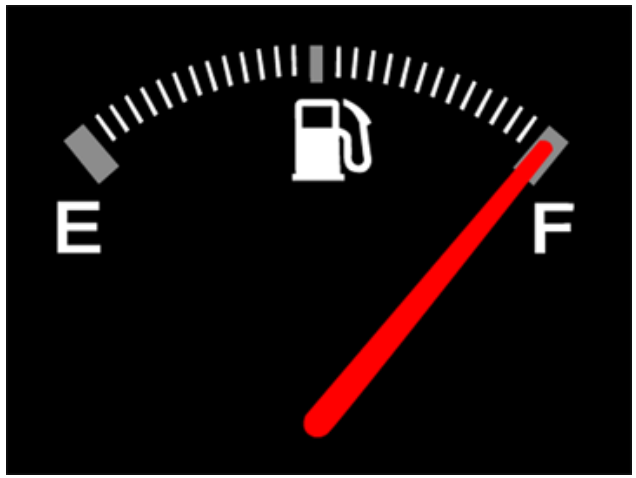
MPC with secret sharing 101

- Each party P_i has $[a] \leftarrow a_i$ s.t. $a = \sum_{i=1}^{n} a_i$.
- Triples generation: $[a] = [b] \cdot [c]$
- Random bits and squares: $[b]$, $[s^2]$.

Preprocessing Phase

MPC with secret sharing 101

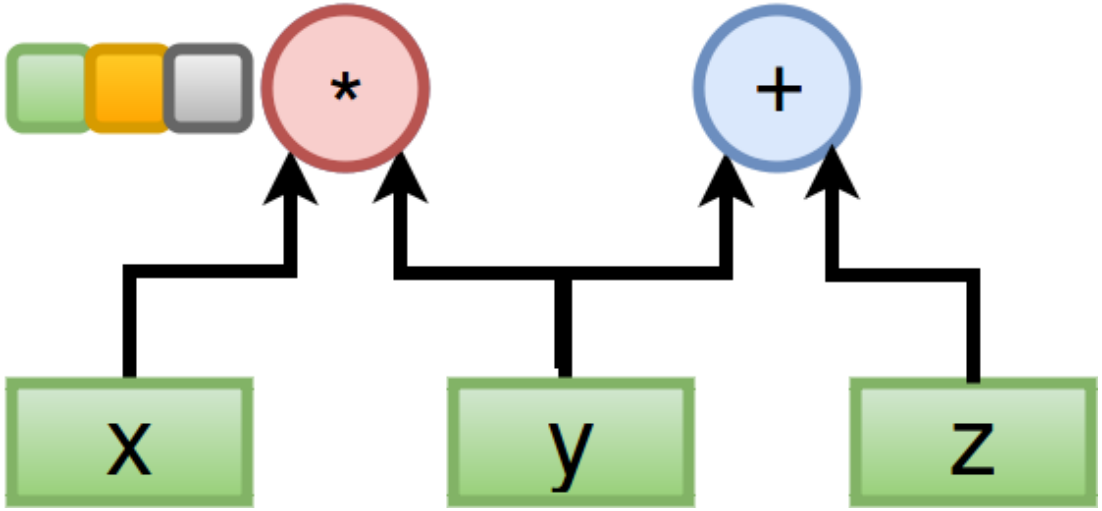
- Use 1 triple for each multiplication gate.
- Number of communcation rounds is given by the circuit depth.



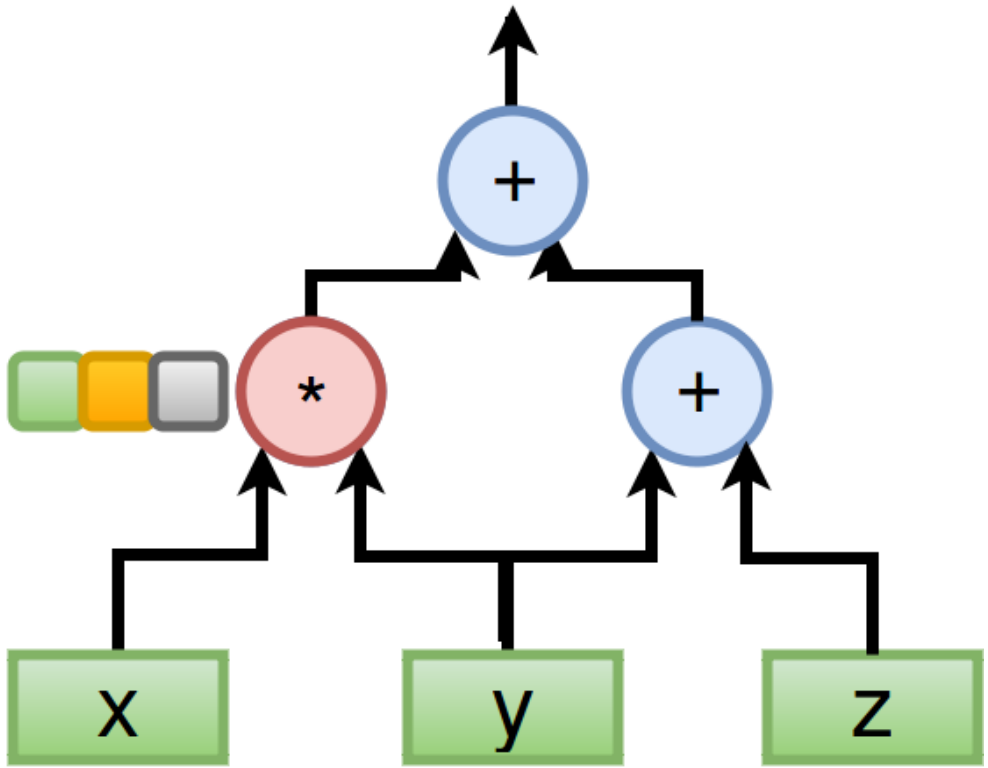
Online Phase

Circuit Evaluation in SPDZ

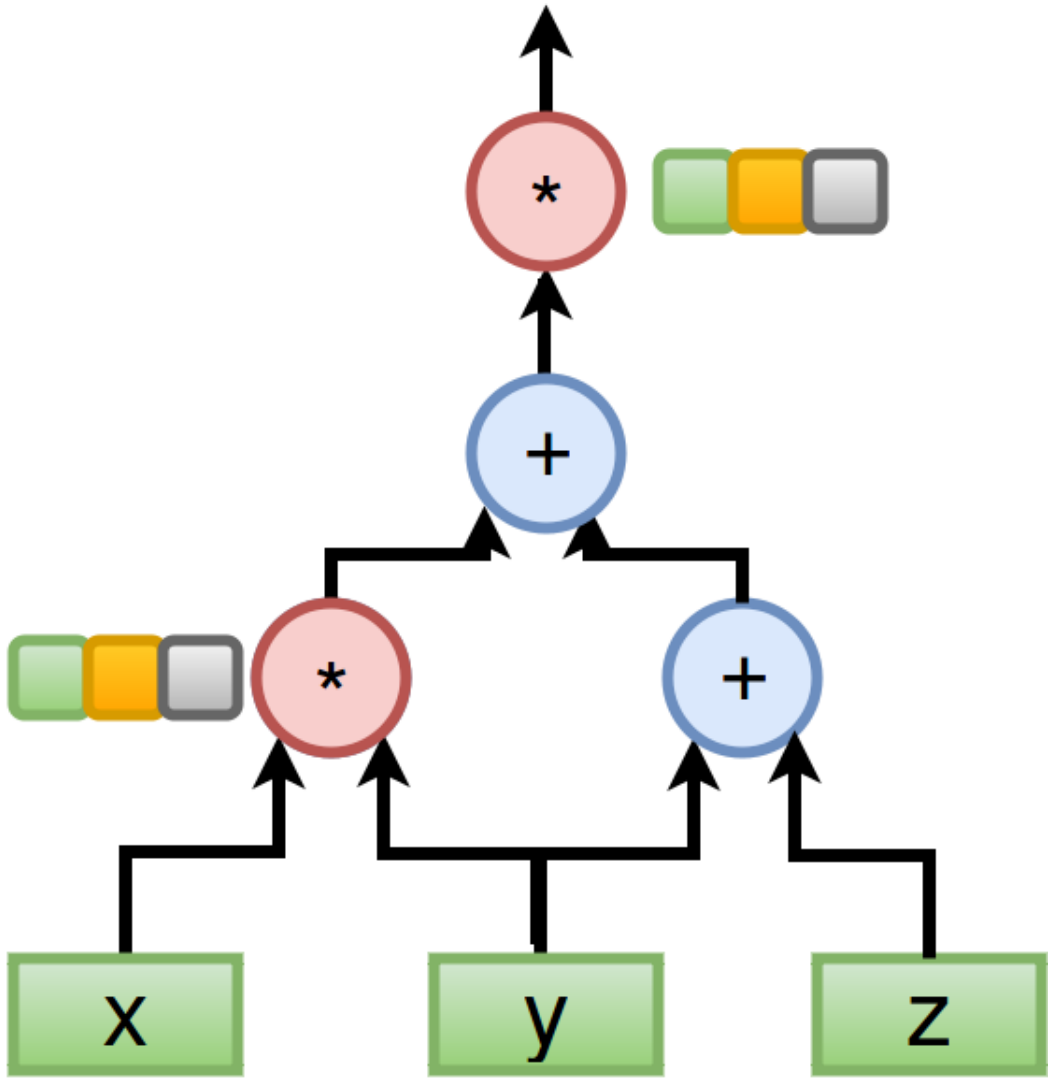
Circuit Evaluation in SPDZ



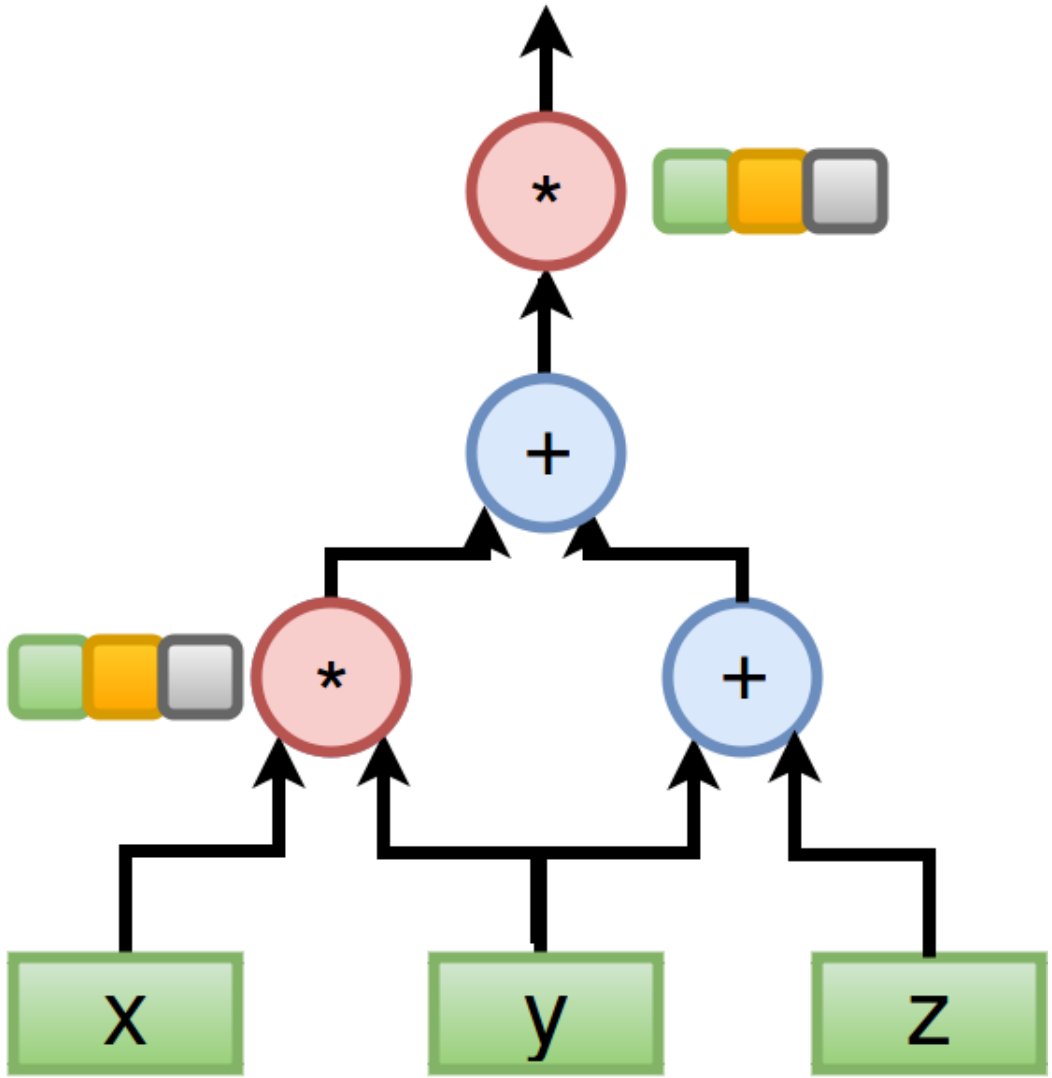
Circuit Evaluation in SPDZ



Circuit Evaluation in SPDZ



Circuit Evaluation in SPDZ



2 triples; 3 rounds.

What PRF's have we looked at?

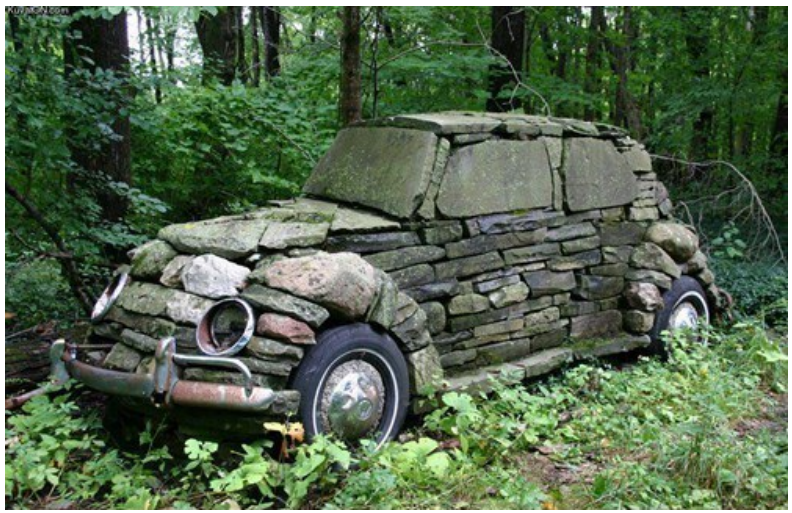
- AES [DR01].
- LowMC (Low Multiplicative Complexity) [ARS+15].
- Naor-Reingold PRF [NR04].
- MiMC (Minimum Multiplicative Complexity) [AGR+16].
- Legendre PRF [Dam88].

Let's play a game

Let's play a game

AES - de-facto benchmark

- 960 multiplications
- 50 rounds



PRF on blocks

AES - de-facto benchmark

- 960 multiplications
- 50 rounds

PRF on blocks

5 ops/s

AES - de-facto benchmark

- 960 multiplications
- 50 rounds

PRF on blocks

8ms latency

AES - de-facto benchmark

- 960 multiplications
- 50 rounds

PRF on blocks

530 ops/s throughput

Naor-Reingold PRF

$$F_{\mathrm{NR}(n)}(\mathbf{k}, \mathbf{x}) = g^{k_0 \cdot \prod_{i=1}^{n} k_i^{x_i}}$$

where $\mathbf{k} = (k_0, \ldots, k_n) \in \mathbb{F}_p^{n+1}$ is the key.

Naor-Reingold PRF

$$F_{\mathrm{NR}(n)}(\mathbf{k}, \mathbf{x}) = g^{k_0 \cdot \prod_{i=1}^{n} k_i^{x_i}}$$

where $\mathbf{k} = (k_0, \ldots, k_n) \in \mathbb{F}_p^{n+1}$ is the key.

Fortunately, in some applications the output must be public!

Naor-Reingold PRF

- Active security version for public output.
- Why EC? Smaller modulus.
- $2 \cdot n$ multiplications.
- $3 + \log n + 1$ rounds.

EC based PRF

Naor-Reingold PRF

- Active security version for public output.
- Why EC? Smaller modulus.
- $4n + 2$ multiplications.
- 7 rounds [CH10].

EC based PRF in constant round

Naor-Reingold PRF

- Active security version for public output.
- Why EC? Smaller modulus.
- $4n + 2$ multiplications.
- 7 rounds [CH10].

EC based PRF in constant round

5 ops/s

Naor-Reingold PRF

- Active security version for public output.
- Why EC? Smaller modulus.
- $4n + 2$ multiplications.
- 7 rounds [CH10].

EC based PRF in constant round

4.3ms latency

Naor-Reingold PRF

- Active security version for public output.
- Why EC? Smaller modulus.
- $4n + 2$ multiplications.
- 7 rounds [CH10].

EC based PRF in constant round

370 ops/s throughput

Naor-Reingold PRF

- Active security version for public output.
- Why EC? Smaller modulus.
- $4n + 2$ multiplications.
- 7 rounds [CH10].

EC based PRF in constant round

Results have shown that over 70% of the time was spent on EC computations.
Computation is the bottleneck, not communication!

MiMC - How does it work?

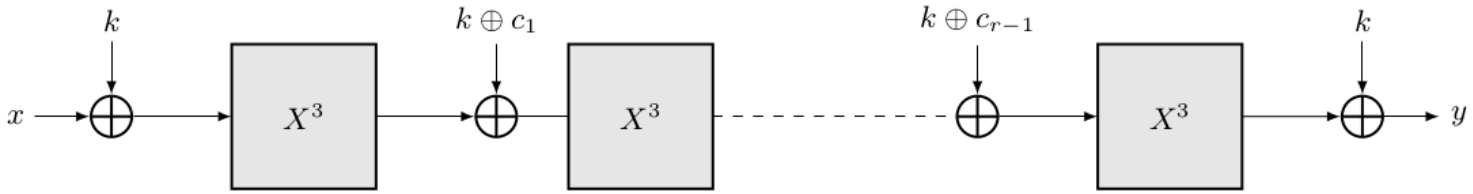


Fig. 1: r rounds of MiMC-n/n

[AGR+16]

MiMC PRF

- 146 multiplications
- 73 rounds
- 1 variant optimized for latency, other for throughput.

MiMC PRF - works in both worlds

MiMC PRF

- 146 multiplications
- 73 rounds
- 1 variant optimized for latency, other for throughput.

MiMC PRF - works in both worlds

34 ops/s

MiMC PRF

- 146 multiplications
- 73 rounds
- 1 variant optimized for latency, other for throughput.

MiMC PRF - works in both worlds

6ms latency

MiMC PRF

- 146 multiplications
- 73 rounds
- 1 variant optimized for latency, other for throughput.

MiMC PRF - works in both worlds

9000 ops/s throughput - **16x** AES

Legendre PRF

In 1988, Damgård conjectured that this sequence is pseuodarandom starting from a random seed k.

$$\left(\frac{k}{p}\right), \left(\frac{k+1}{p}\right), \left(\frac{k+2}{p}\right), \ldots$$

Legendre PRF - 1 bit output

- $\log p$ multiplications.
- $\log p$ rounds.

Legendre PRF - old version

Legendre PRF - 1 bit output

- ~~$\log p$~~ 2 multiplications.
- ~~$\log p$~~ 3 rounds.

Legendre PRF - new version

Legendre PRF - 1 bit output

- ~~$\log p$~~ 2 multiplications.
- ~~$\log p$~~ 3 rounds.

Legendre PRF - new version

1225 ops/s - **250x** AES

Legendre PRF - 1 bit output

- ~~$\log p$~~ 2 multiplications.
- ~~$\log p$~~ 3 rounds.

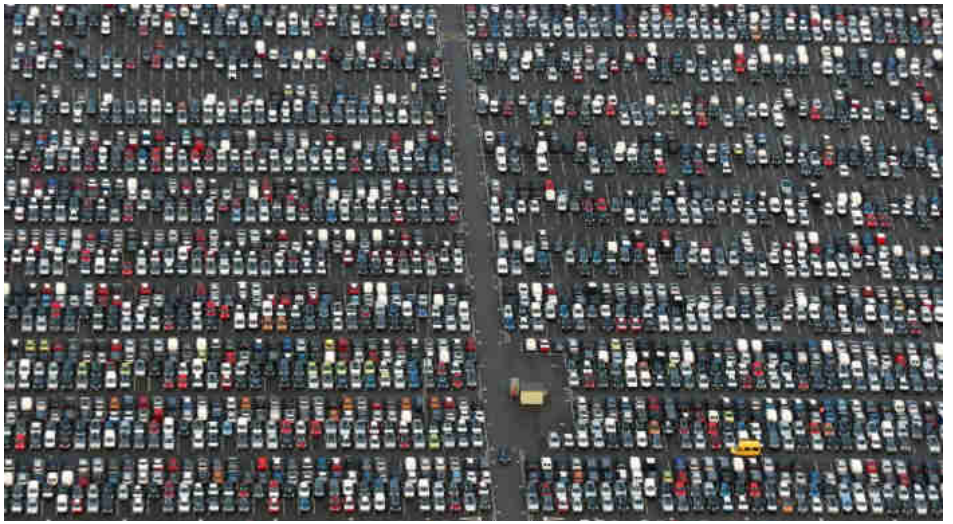
Legendre PRF - new version

0.3ms latency - **25x** faster AES

Legendre PRF - 1 bit output

- ~~$\log p$~~ 2 multiplications.
- ~~$\log p$~~ 3 rounds.

Legendre PRF - new version



202969 ops/s throughput - **380x** AES

Security of Legendre PRF

Is it secure?

Security of Legendre PRF

Is it secure?

Yes, we give a reduction to the SLS problem.

How does it work?

Protocol Π_{Legendre}

Let α be a fixed, quadratic non-residue modulo p, i.e. $\left(\frac{\alpha}{p}\right) = -1$.

Eval: To evaluate $F_{\text{Leg(bit)}}$ on input $[x]$ with key $[k]$:

1. Take a random square $[s^2]$ and a random bit $[b]$
2. $[t] \leftarrow [s^2] \cdot ([b] + \alpha \cdot (1 - [b]))$
3. $u \leftarrow \text{Open}([t] \cdot ([k] + [x]))$
4. Output $[y] \leftarrow \left(\frac{u}{p}\right) \cdot (2[b] - 1)$

Securely computing the $F_{\text{Leg(bit)}}$ PRF with shared output

How does it work?

Protocol Π_{Legendre}

Let α be a fixed, quadratic non-residue modulo p, i.e. $\left(\frac{\alpha}{p}\right) = -1$.

Eval: To evaluate $F_{\text{Leg(bit)}}$ on input $[x]$ with key $[k]$:

1. Take a random square $[s^2]$ and a random bit $[b]$
2. $[t] \leftarrow [s^2] \cdot (\boxed{[1]} + \alpha \cdot (1 - \boxed{[1]}))$
3. $u \leftarrow \text{Open}([s^2] \cdot ([k] + [x]))$
4. Output $[y] \leftarrow \left(\frac{u}{p}\right) \cdot (2\boxed{[1]} - 1)$

Securely computing the $F_{\text{Leg(bit)}}$ PRF with shared output

How does it work?

Protocol Π_{Legendre}

Let α be a fixed, quadratic non-residue modulo p, i.e. $\left(\frac{\alpha}{p}\right) = -1$.

Eval: To evaluate $F_{\text{Leg(bit)}}$ on input $[x]$ with key $[k]$:

1. Take a random square $[s^2]$ and a random bit $[b]$
2. $[t] \leftarrow [s^2] \cdot ([0] + \alpha \cdot (1 - [0]))$
3. $u \leftarrow \text{Open}([s^2\alpha] \cdot ([k] + [x]))$
4. Output $[y] \leftarrow \left(\frac{u}{p}\right) \cdot (2 [0] - 1)$

Securely computing the $F_{\text{Leg(bit)}}$ PRF with shared output

Summary

- PRF's mod p in MPC are fast! Can you find other applications built on top of these?
- Using special preprocessed data can help you often in SPDZ online phase.
- For proofs, WAN timings, other details, check out our paper!

Thank you!

Thank you!

PRF	Best latency	Best throughput		Prep. (ops/s)	Cleartext (ops/s)
	(ms/op)	Batch size	ops/s		
AES	8.378	2048	552	5.097	106268670
$F_{\mathsf{NR}(128)}(\mathsf{log})$	4.375	1024	370	4.787	1359
$F_{\mathsf{NR}(128)}(\mathsf{const})$	4.549	256	281	2.384	1359
$F_{\mathsf{Leg}(\mathsf{bit})}$	0.393	1024	163400	1225	17824464
$F_{\mathsf{Leg}(1)}$	1.418	64	1331	9.574	115591
$F_{\mathsf{MiMC}}(\mathsf{basic})$	14.053	1024	6561	33.575	189525
$F_{\mathsf{MiMC}}(\mathsf{cube})$	7.267	2048	5536	33.575	189525

Table 3. Performance of the PRFs in a LAN setting

WAN results

PRF	Best latency	Best throughput		Prep. (ops/s)
	(ms/op)	Batch size	ops/s	
AES	2640	1024	31.947	0.256
$F_{\mathsf{NR}(128)}(\mathsf{log})$	713	1024	59.703	0.2359
$F_{\mathsf{NR}(128)}(\mathsf{const})$	478	1024	30.384	0.1175
$F_{\mathsf{Leg}(\mathsf{bit})}$	202	1024	2053	60.241
$F_{\mathsf{Leg}(1)}$	210	512	68.413	0.4706
$F_{\mathsf{MiMC}}(\mathsf{basic})$	7379	512	59.04	1.650
$F_{\mathsf{MiMC}}(\mathsf{cube})$	3691	512	79.66	1.650

Table 4. Performance of the PRFs in a simulated WAN setting