SWIFT: Super-fast and Robust Privacy-Preserving Machine Learning

Nishat Koti, Mahak Pancholi, Arpita Patra, **Ajith Suresh**

Indian Institute of Science, Bangalore

30th USENIX Security Symposium

*Full Version: https://eprint.iacr.org/2020/592

Acknowledgements: Nishat Koti – Cisco PhD Fellowship, Mahak Pancholi - Cisco MTech Fellowship, Ajith Suresh - Google PhD Fellowship, Arpita Patra - SERB MATRICS (Theoretical Sciences) Grant 2020 and Google India AI/ML Research Award 2020.

Privacy in Machine Learning

- Machine Learning (ML)
  - Automobile, Healthcare, Finance, …..

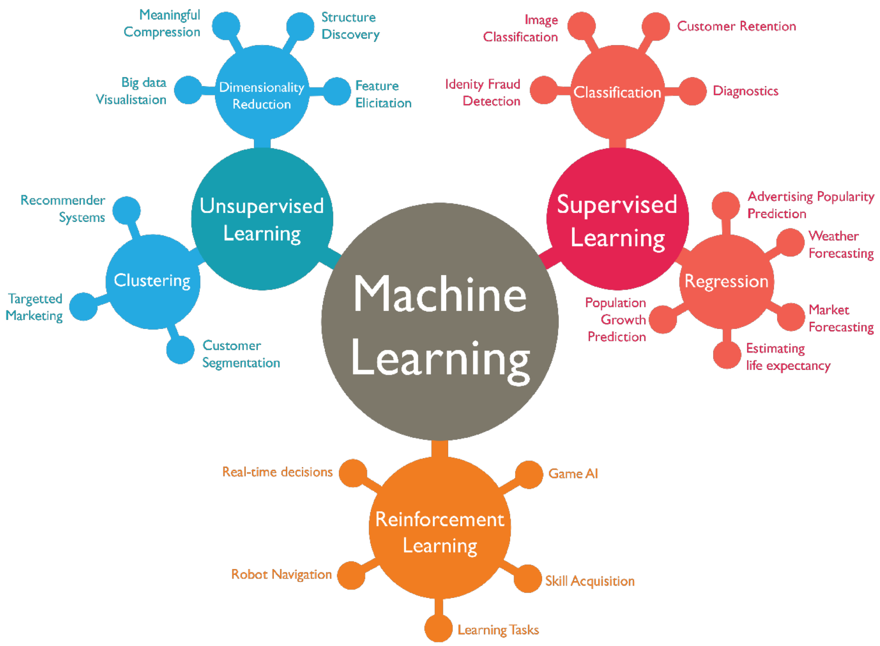


Image credits: Abdul Wahid

Privacy in Machine Learning

- Machine Learning (ML)
  - Automobile, Healthcare, Finance, …..
- Phases: Training & Inference

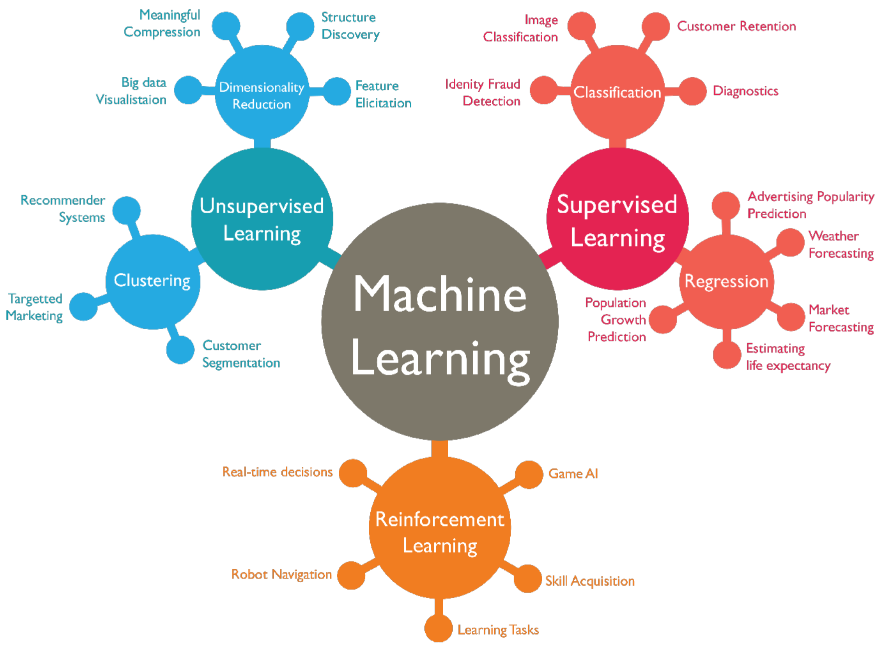
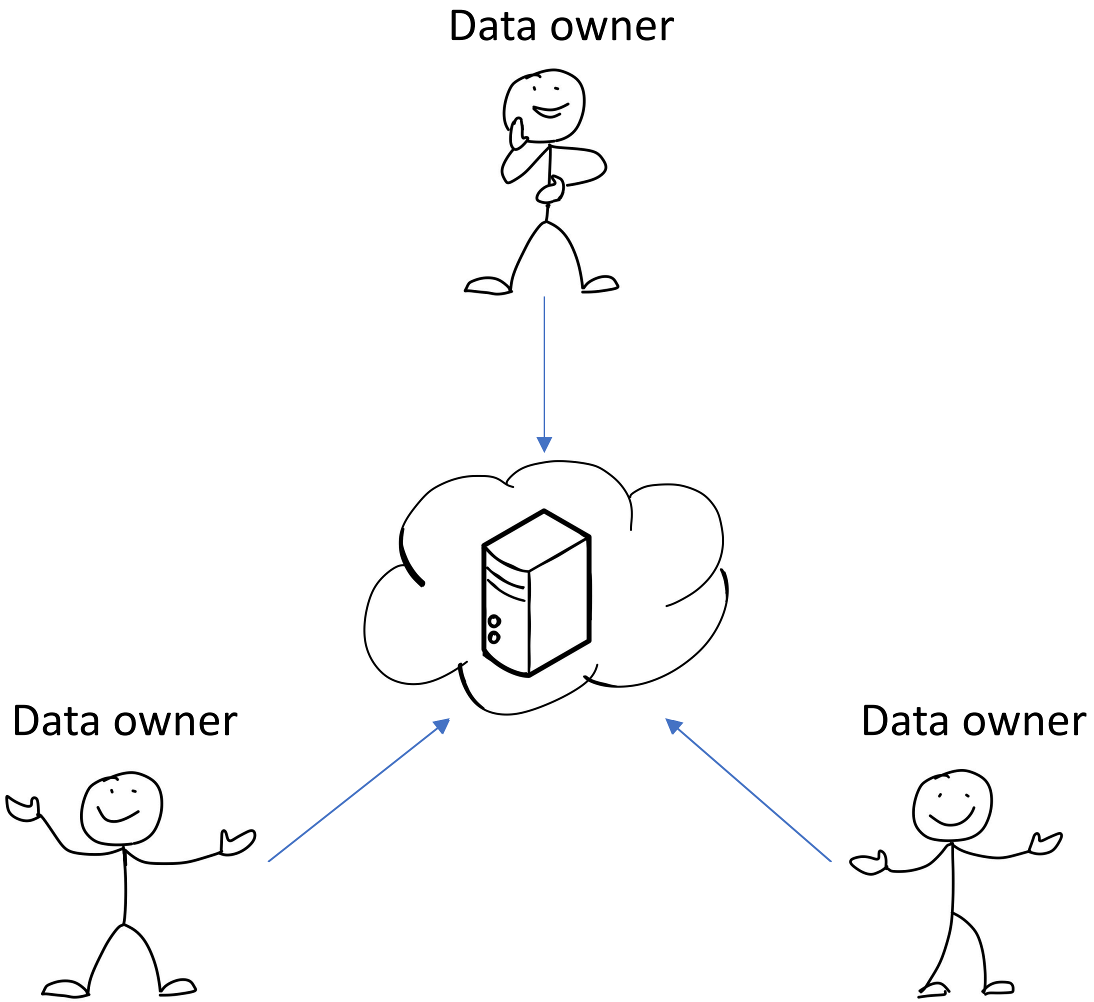
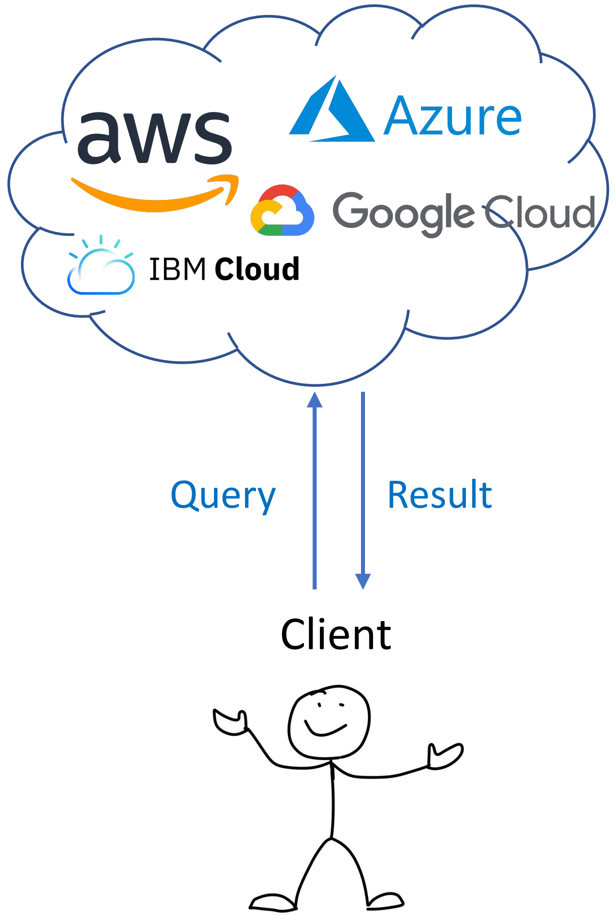


Image credits: Abdul Wahid

Privacy in Machine Learning



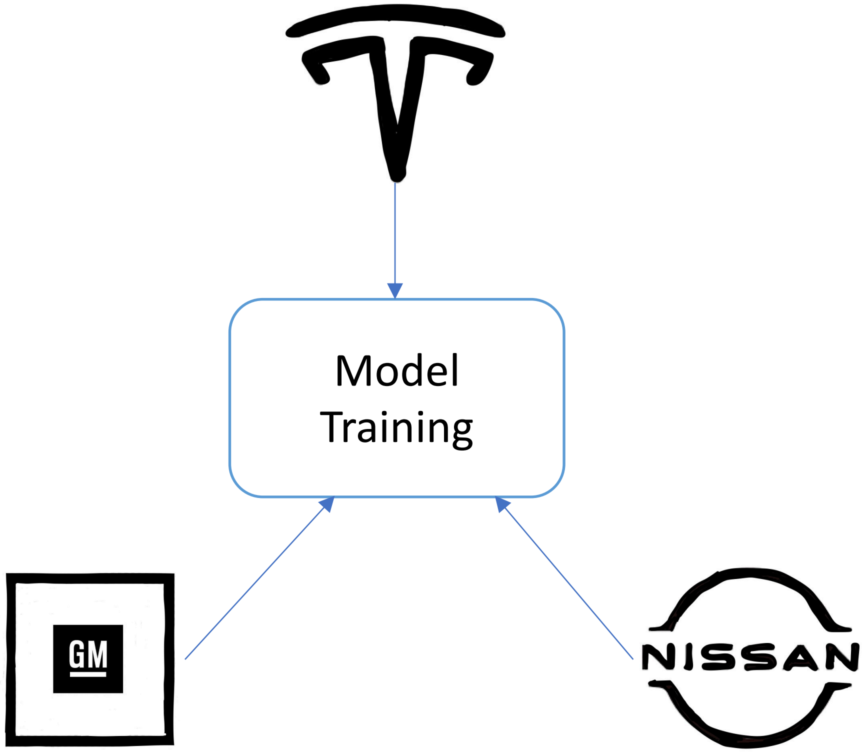
ML Training



ML Inference

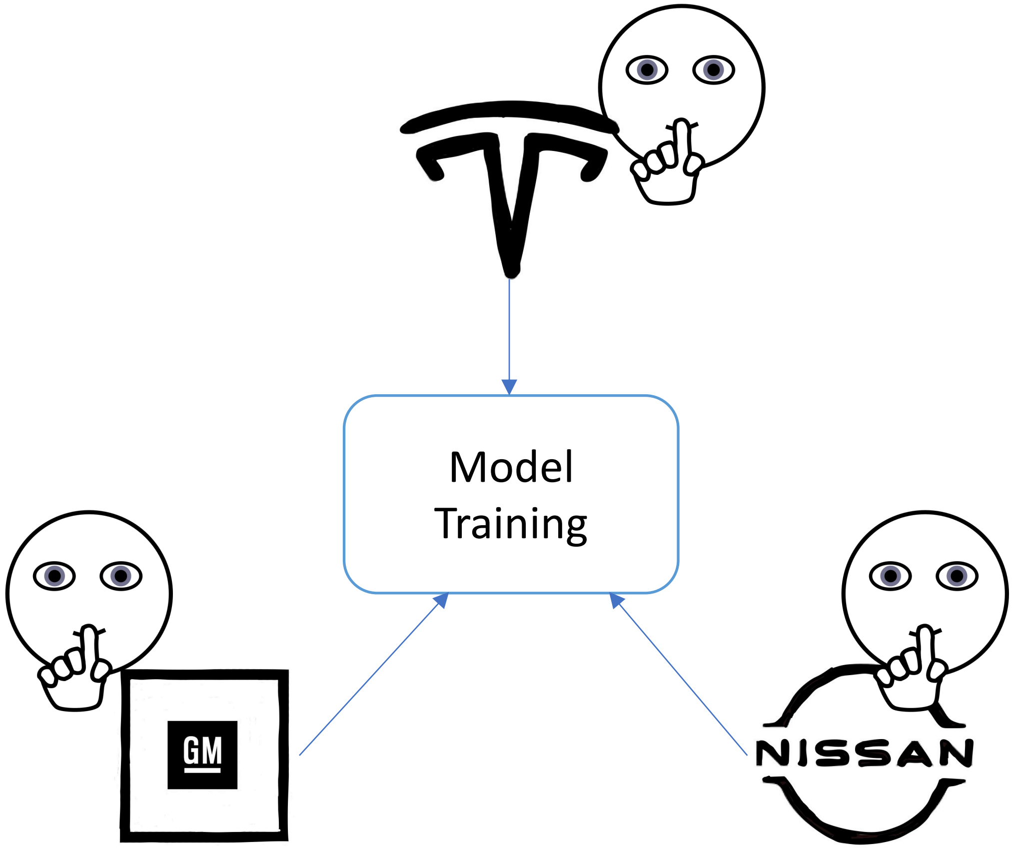
Privacy in Machine Learning

- Machine Learning (ML)
  - Automobile, Healthcare, Finance, ...
- Phases: Training & Inference
- More the data -> better the model



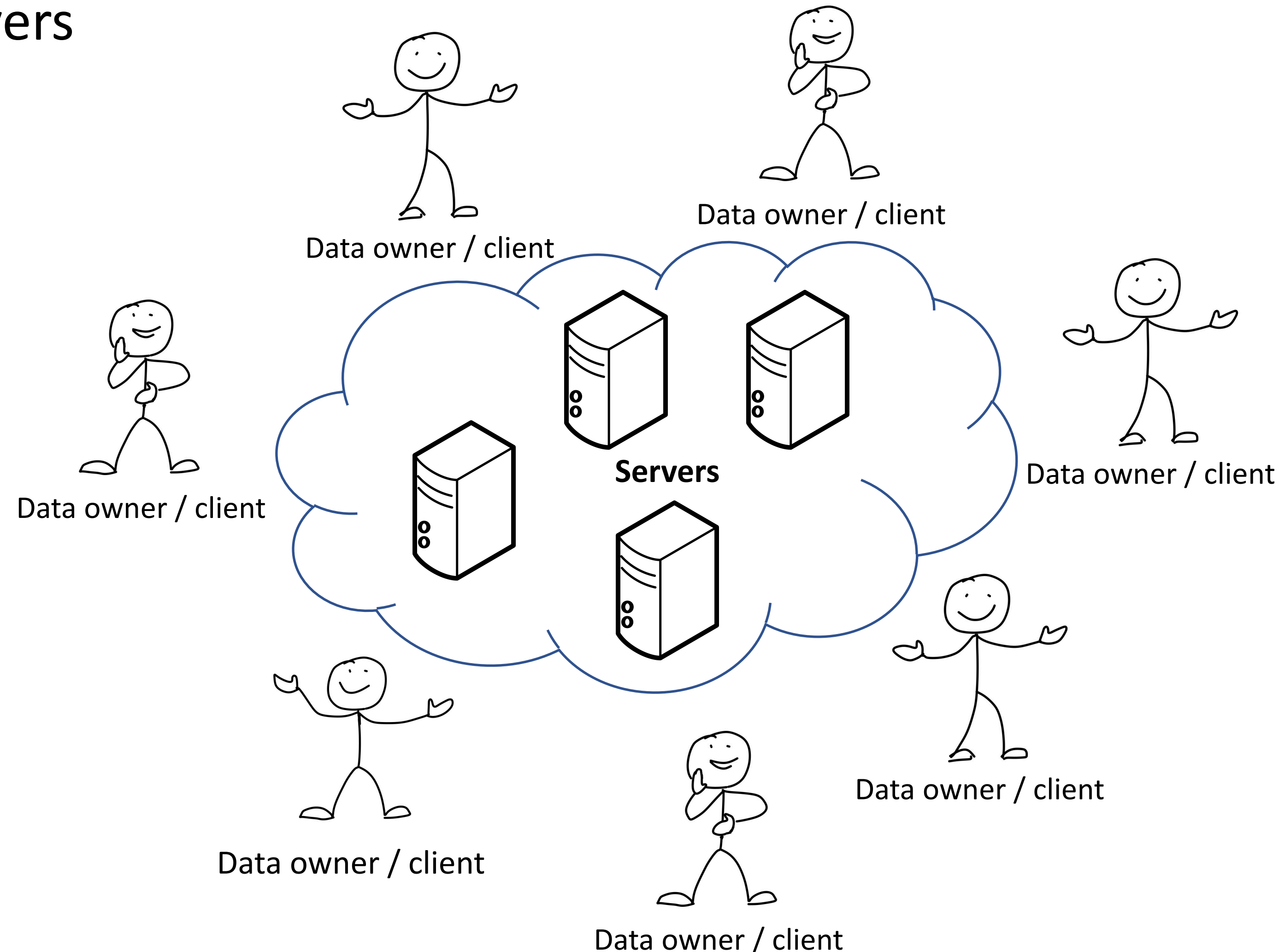
Privacy in Machine Learning

- Machine Learning (ML)
  - Automobile, Healthcare, Finance, …..
- Phases: Training & Inference
- More the data -> better the model
- **Privacy of the data ??**



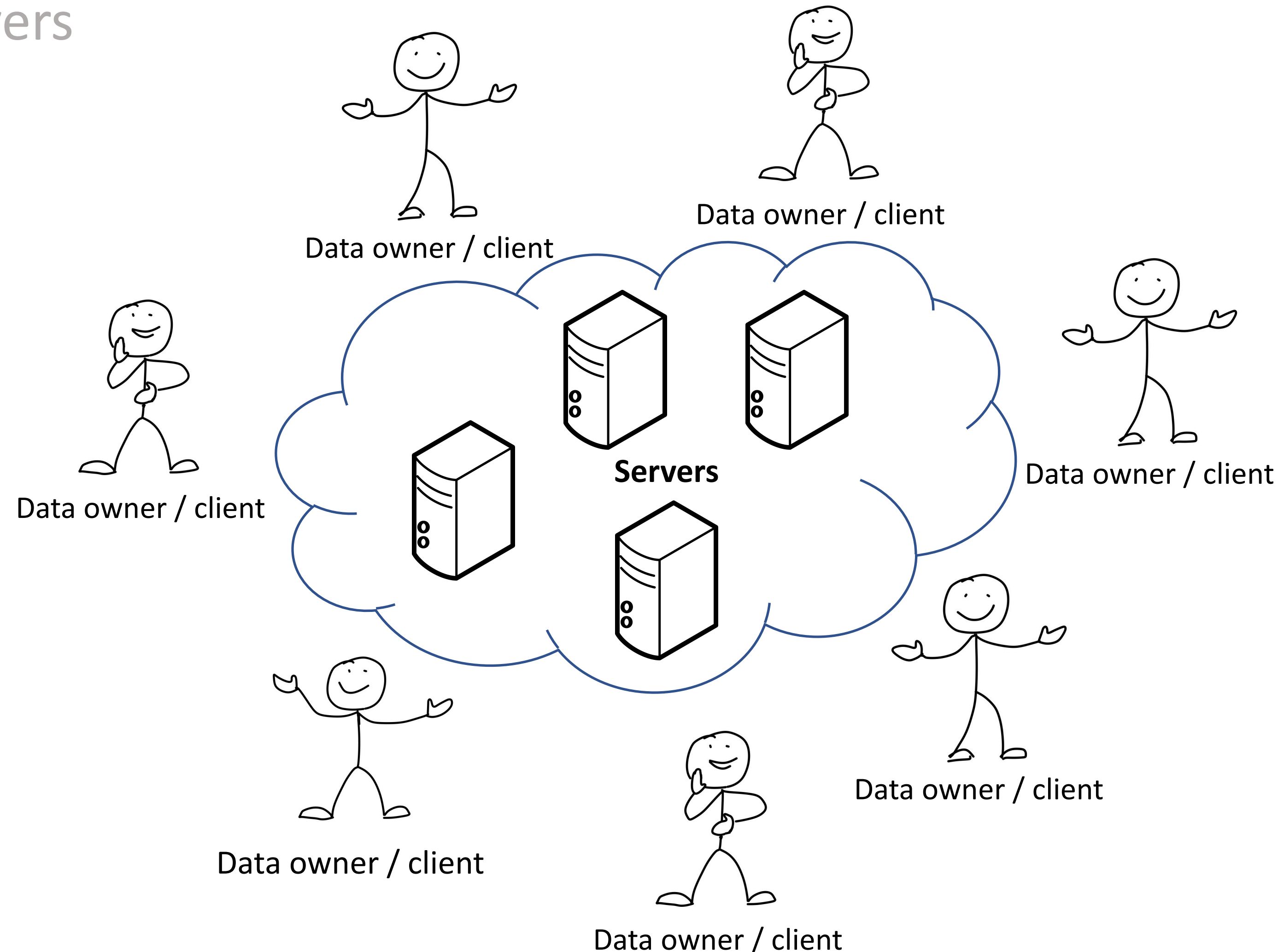
Secure Outsourced Computation (SOC)

- Computation -> hired powerful servers



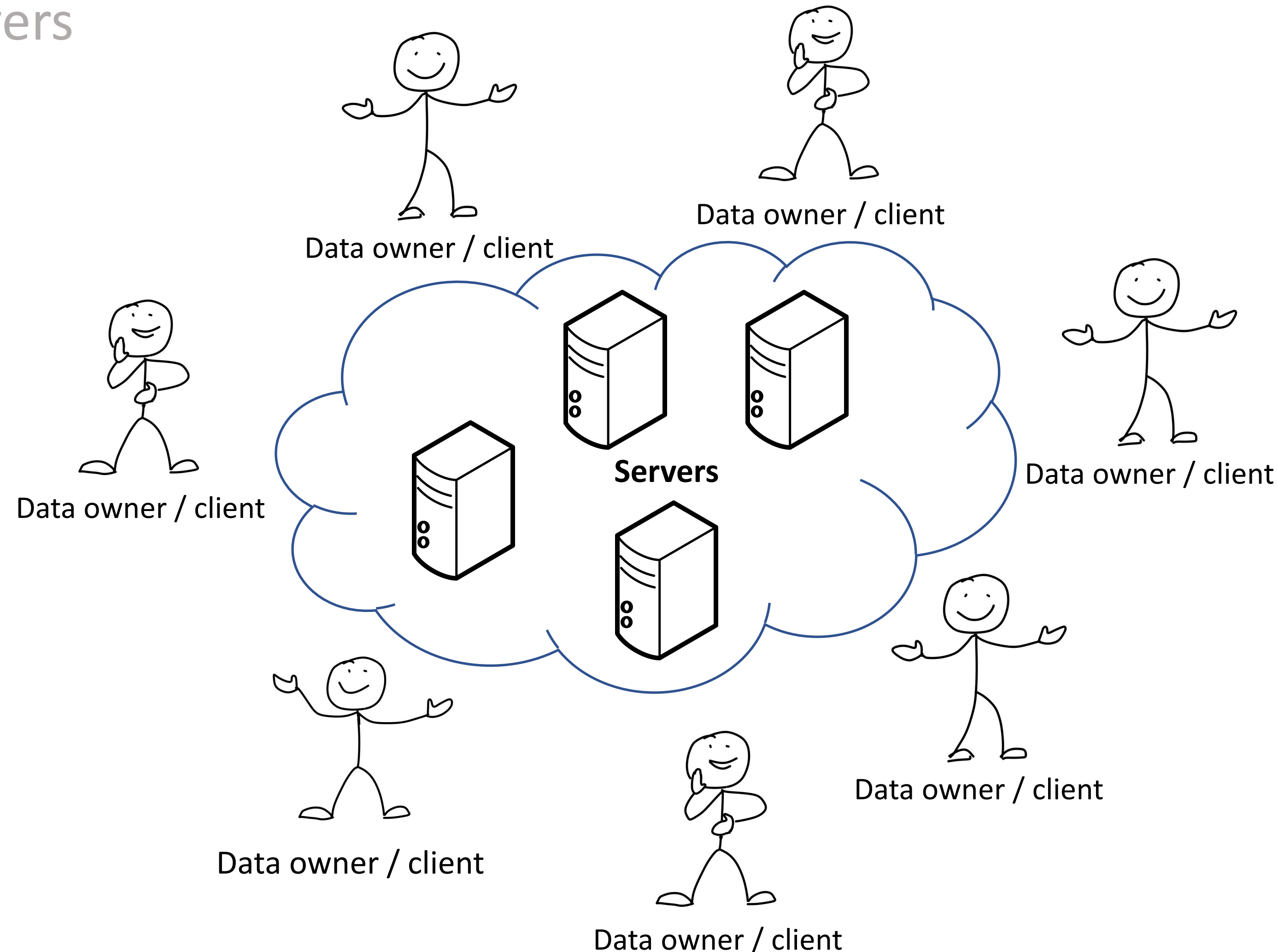
Secure Outsourced Computation (SOC)

- Computation -> hired powerful servers
- Average end users benefit



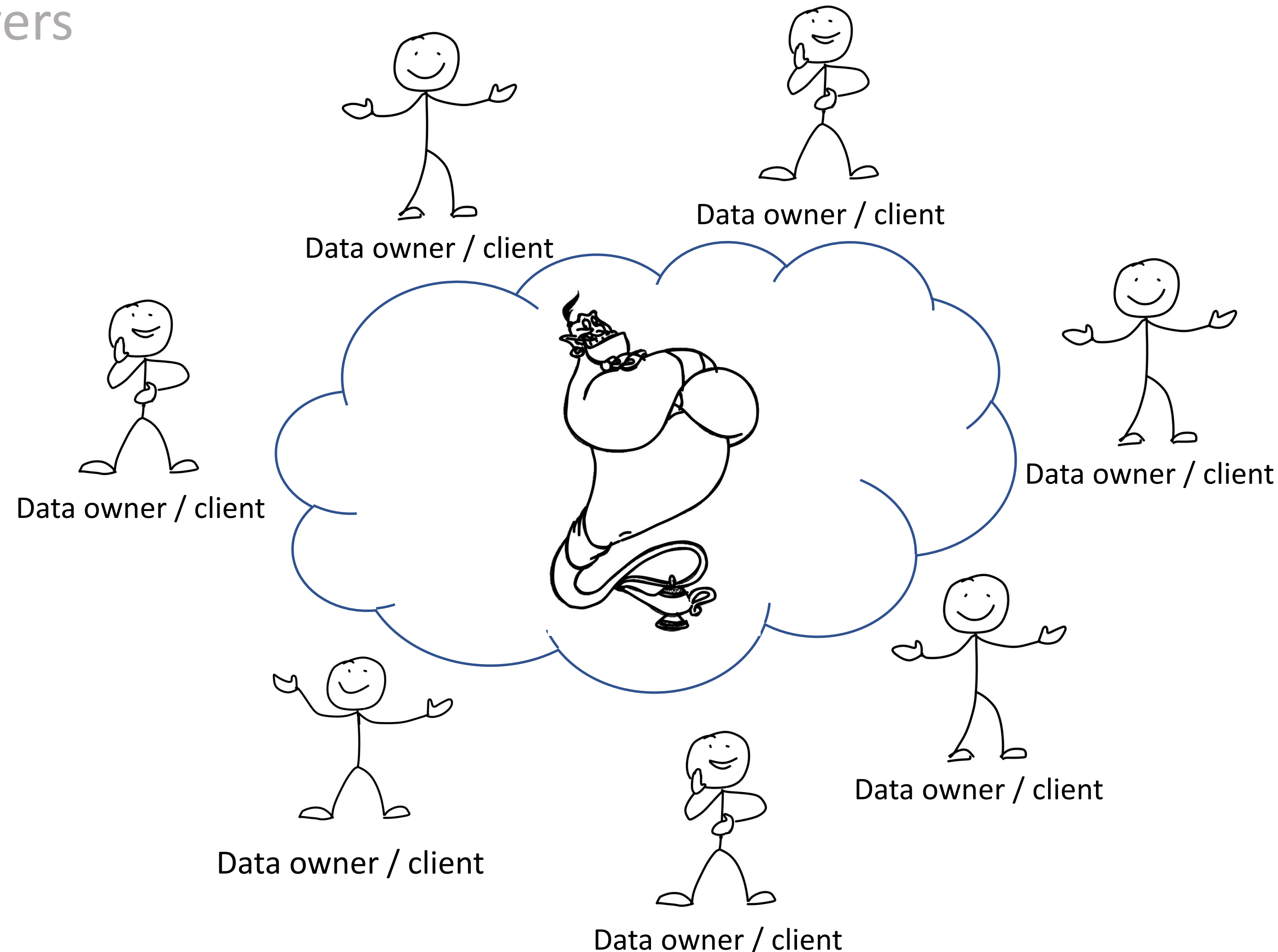
Secure Outsourced Computation (SOC)

- Computation -> hired powerful servers
- Average end users benefit
- Solves issue for large population



Secure Outsourced Computation (SOC)

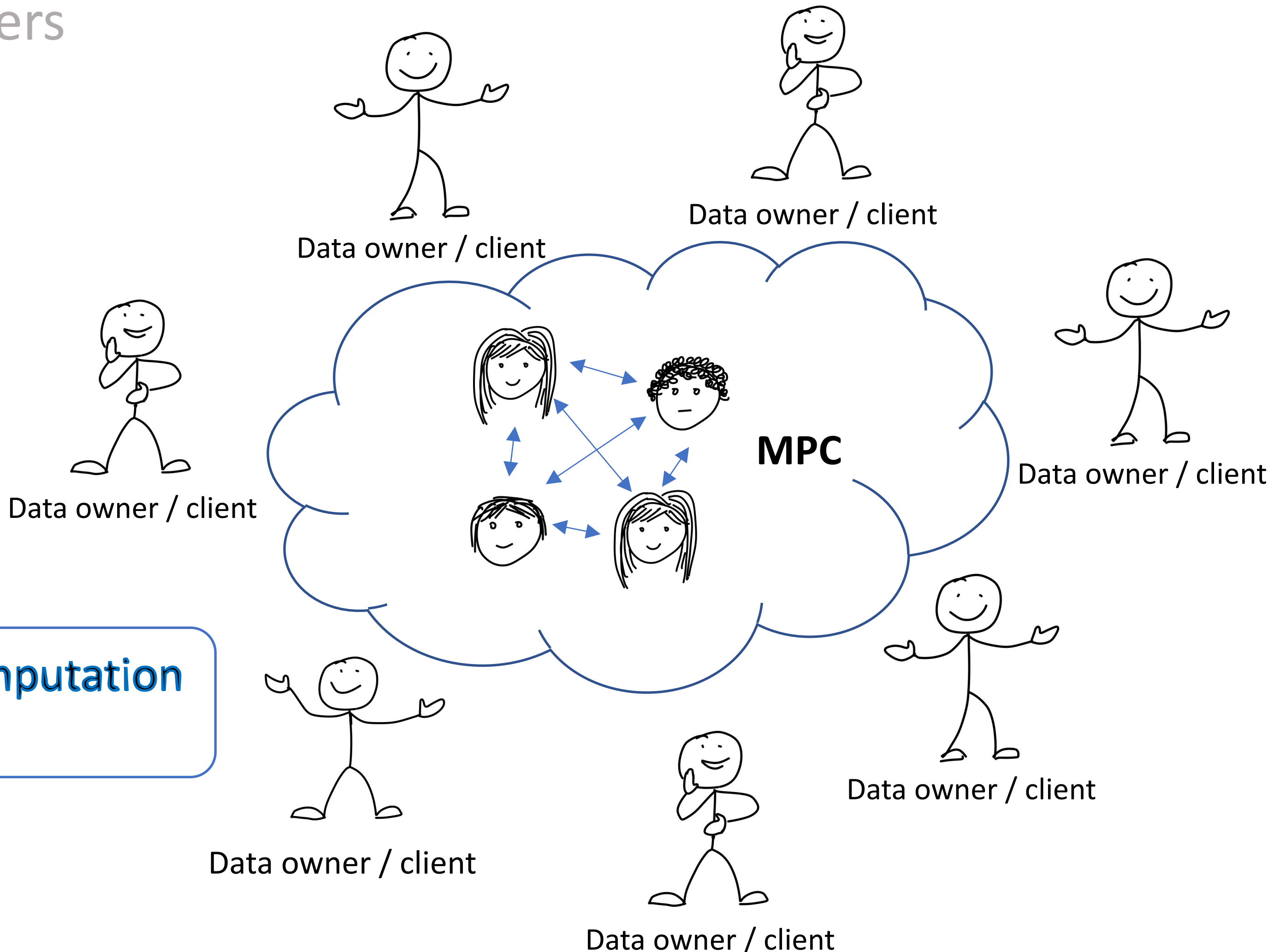
- Computation -> hired powerful servers
- Average end users benefit
- Solves issue for large population
- Desired properties:
  - Privacy
  - Efficiency
  - Robustness



Secure Outsourced Computation (SOC)

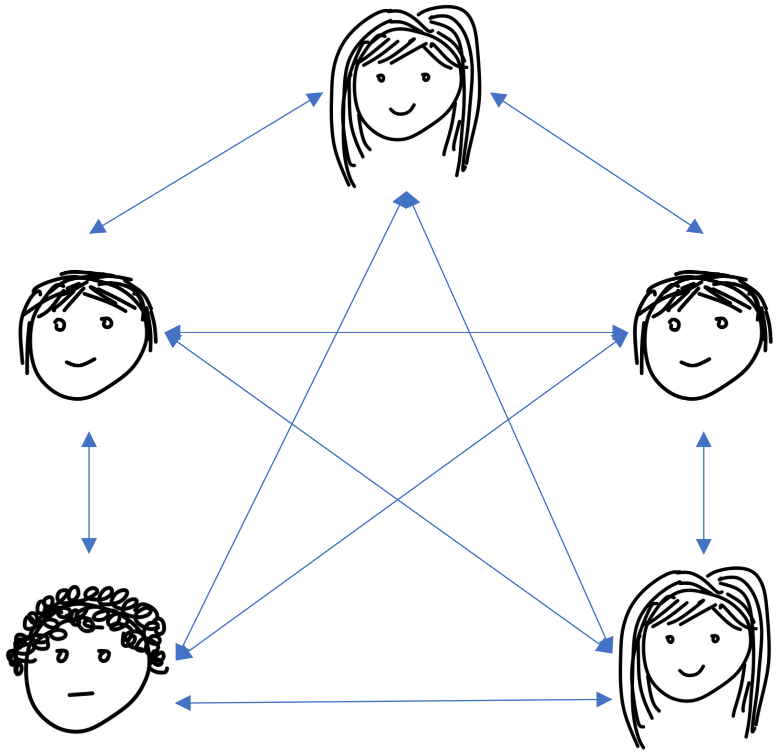
- Computation -> hired powerful servers
- Average end users benefit
- Solves issue for large population
- Desired properties:
  - Privacy
  - Efficiency
  - Robustness

Secure Multiparty Computation (MPC)



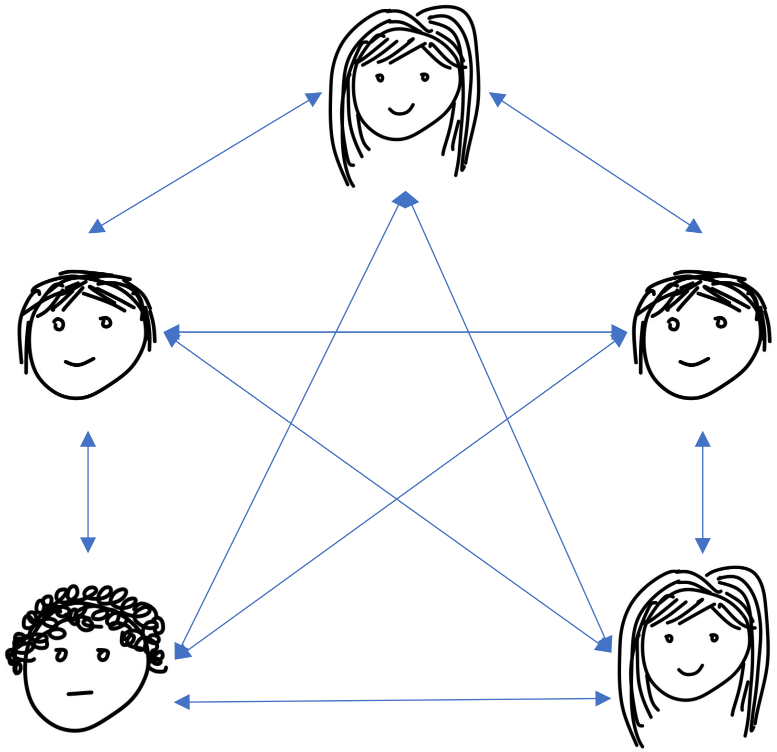
Secure Multiparty Computation [Yao'82]

- MPC allows a set of parties with private inputs to compute some joint function of their inputs



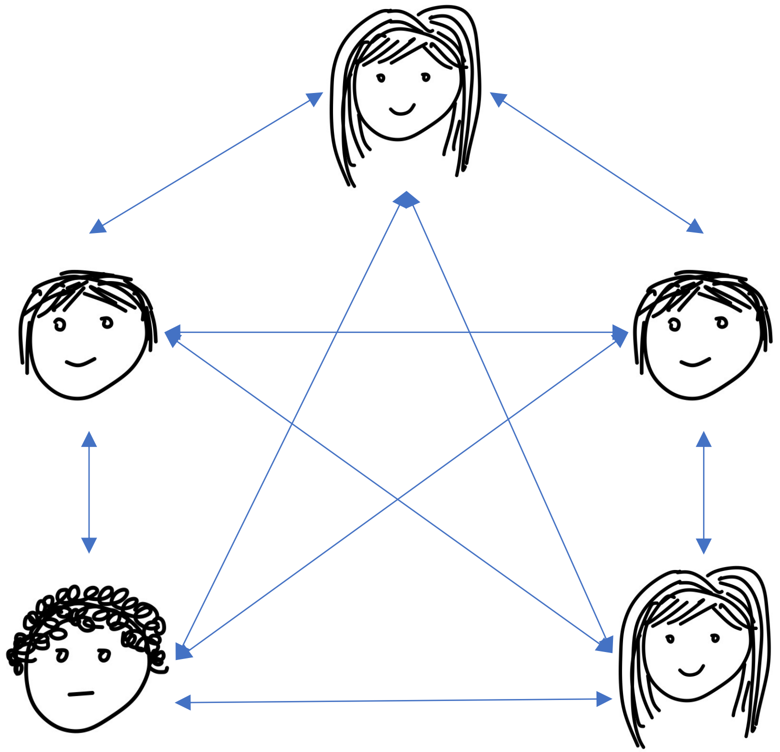
Secure Multiparty Computation [Yao'82]

- MPC allows a set of parties with private inputs to compute some joint function of their inputs
- Properties of MPC:
  - **Correctness** – Parties should obtain the correct function output



Secure Multiparty Computation [Yao'82]

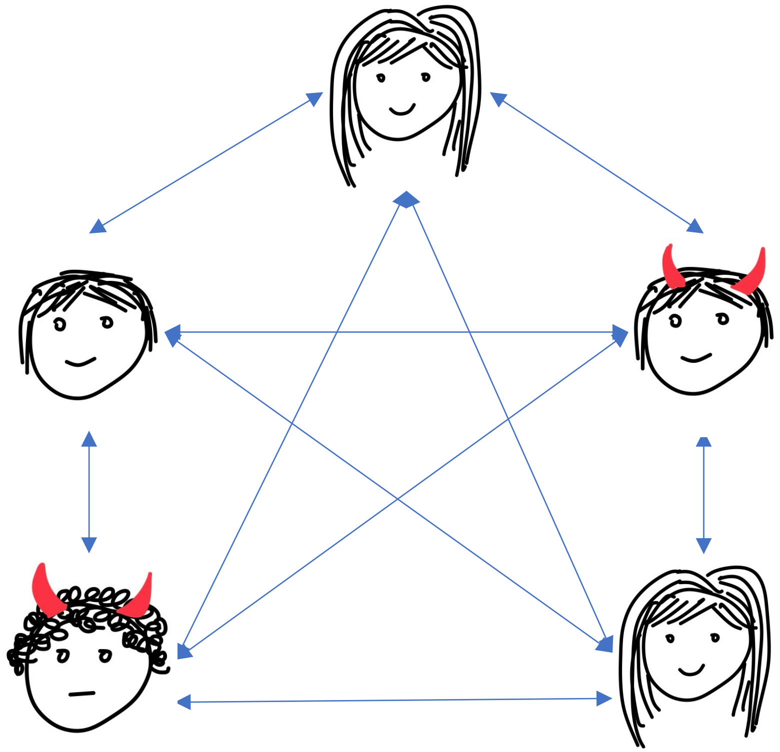
- MPC allows a set of parties with private inputs to compute some joint function of their inputs
- Properties of MPC:
  - **Correctness** – Parties should obtain the correct function output
  - **Privacy** – Nothing more than the function output should be revealed



Secure Multiparty Computation [Yao'82]

- Adversary

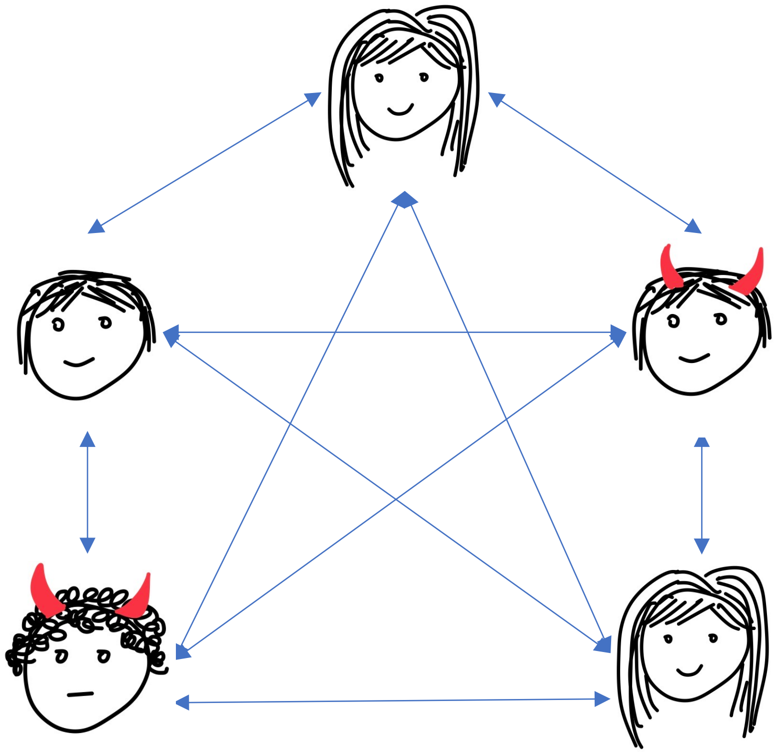
  - Semi-honest vs **Malicious**



Secure Multiparty Computation [Yao'82]

- Adversary

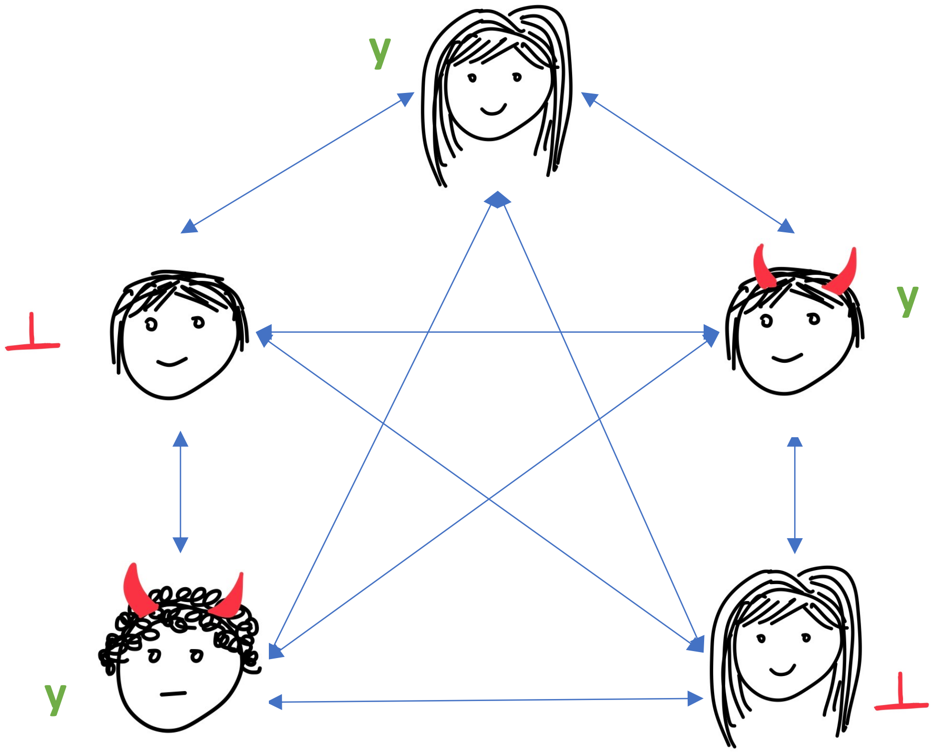
  - Semi-honest vs **Malicious**
  - **Honest majority** vs Dishonest majority



Secure Multiparty Computation [Yao'82]

- Security

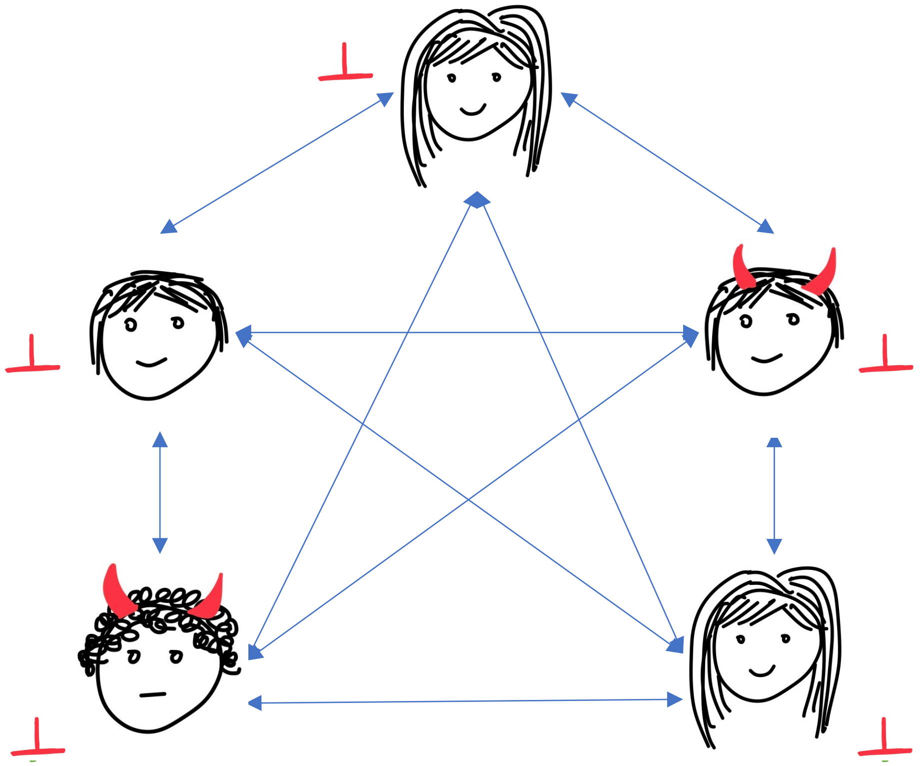
  - Abort



Secure Multiparty Computation [Yao'82]

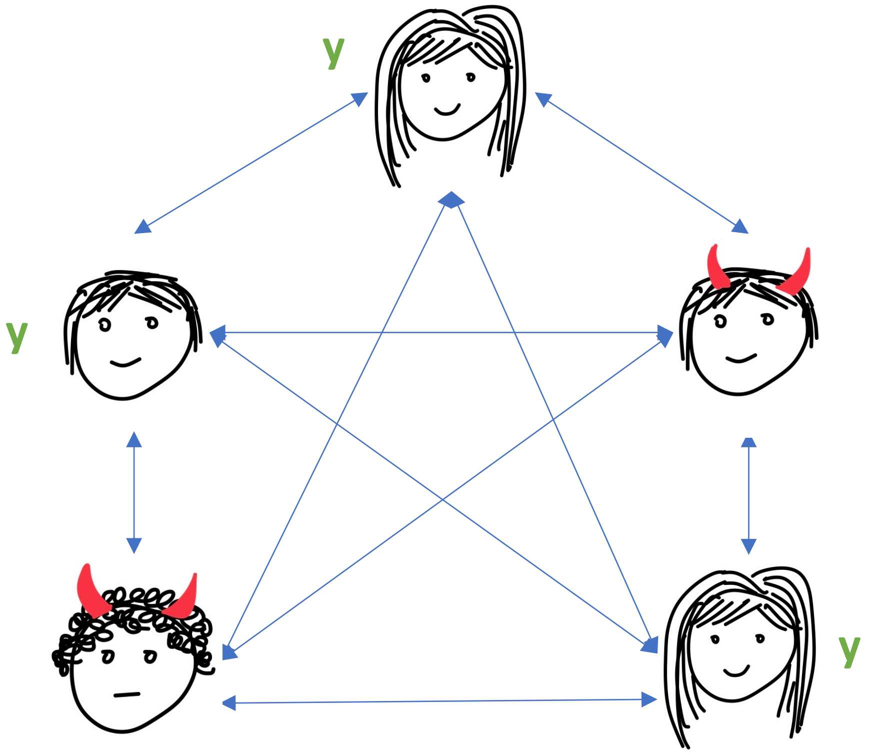
- Security

  - Abort
  - Fairness



Secure Multiparty Computation [Yao'82]

- Security
  - Abort
  - Fairness
  - **Guaranteed Output Delivery (GOD)**

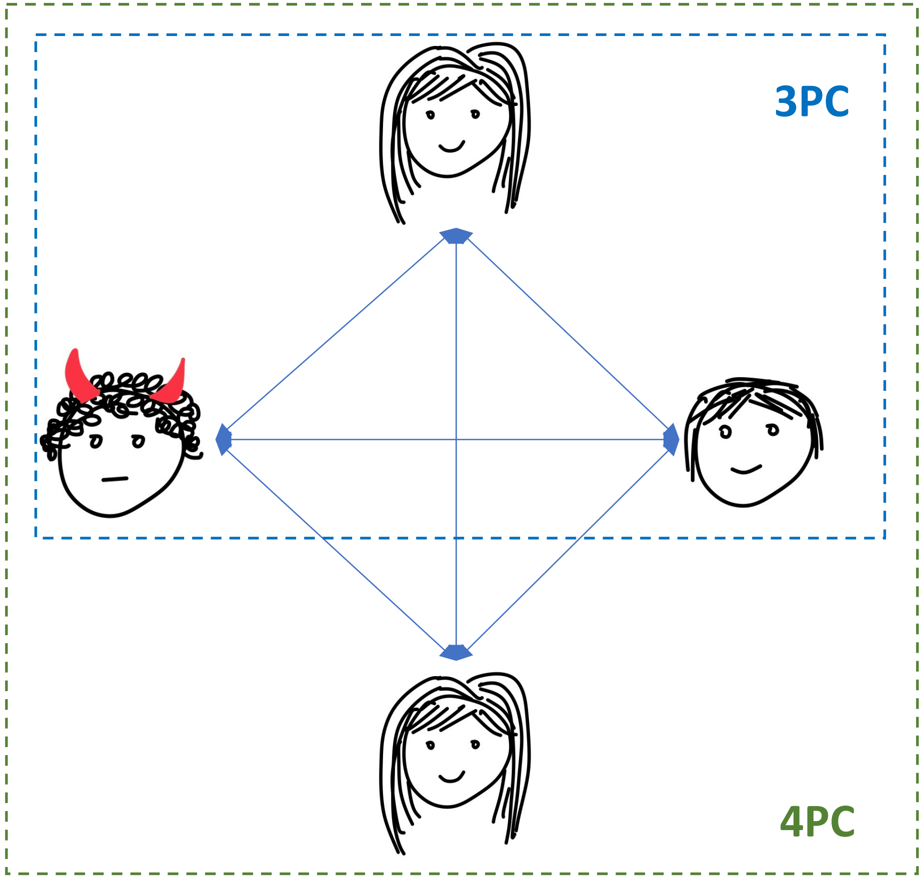


SWIFT Protocol

SWIFT Protocol

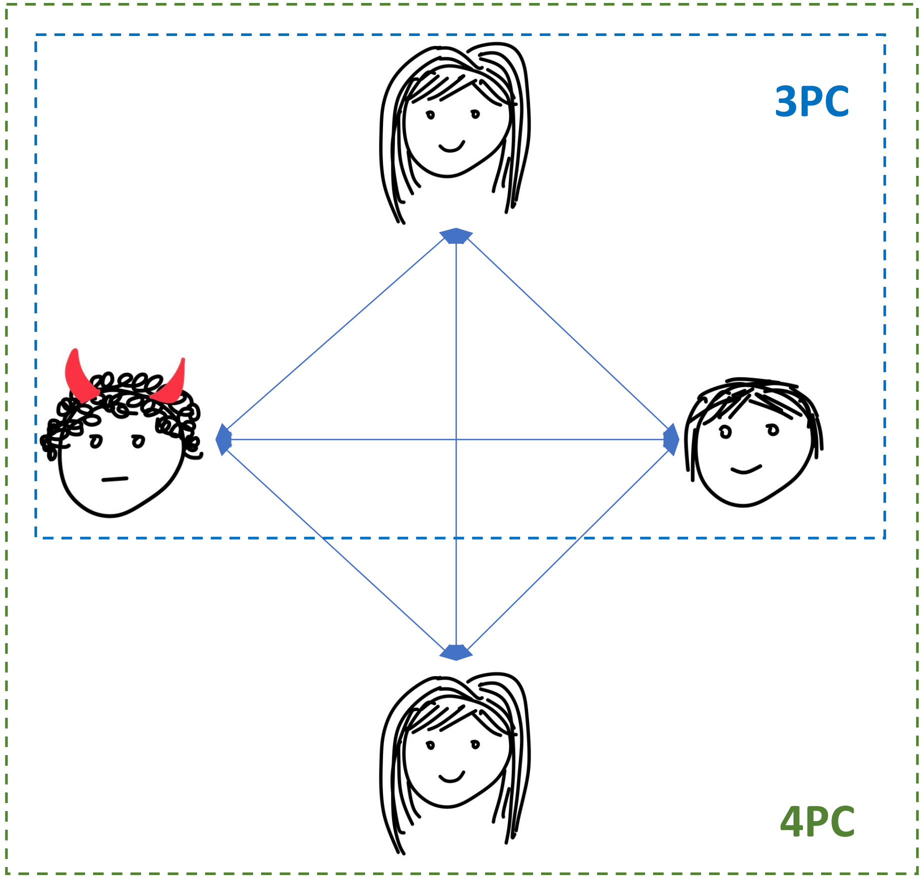
- 3PC and 4PC protocols for malicious corruption

❑ MPC for small population
❑ efficiency and simplicity



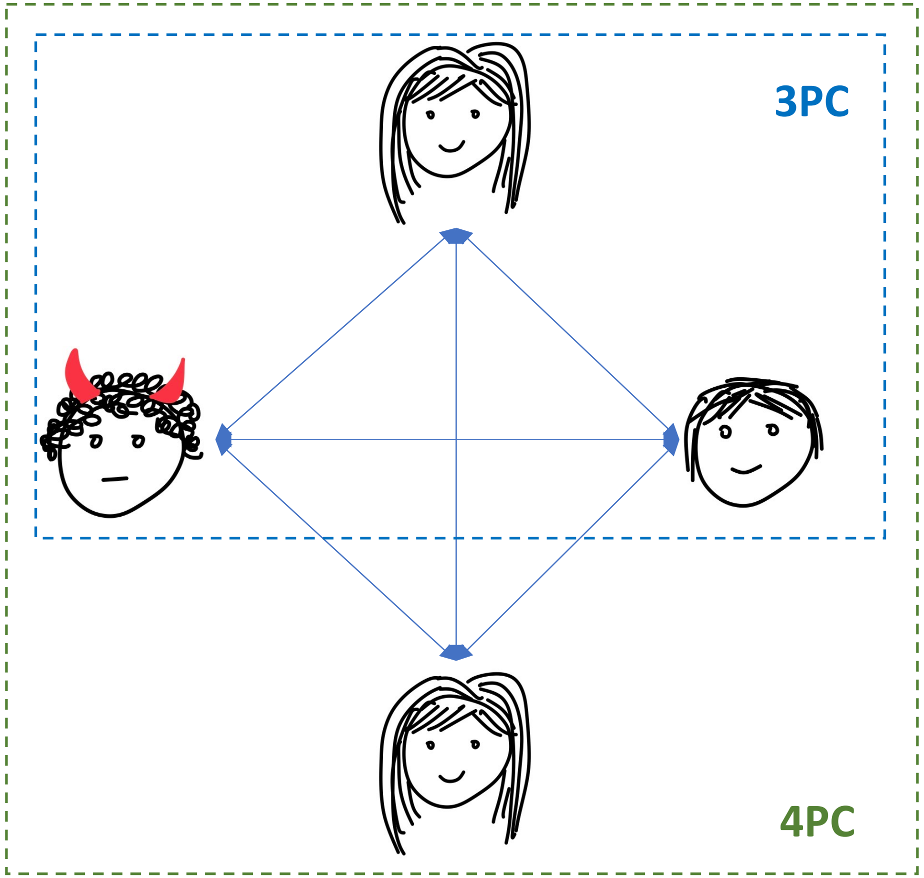
SWIFT Protocol

- 3PC and 4PC protocols for malicious corruption
- Honest-majority setting
  - at most 1 corruption



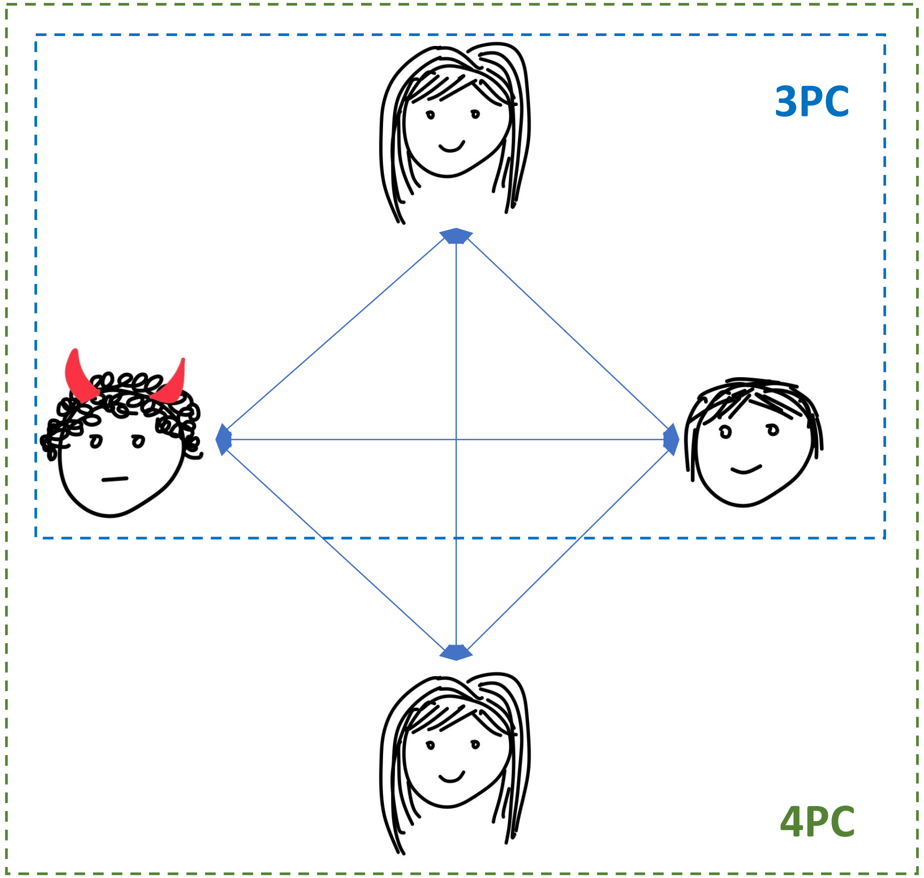
SWIFT Protocol

- 3PC and 4PC protocols for malicious corruption
- Honest-majority setting
  - at most 1 corruption
- Guaranteed Output Delivery



SWIFT Protocol

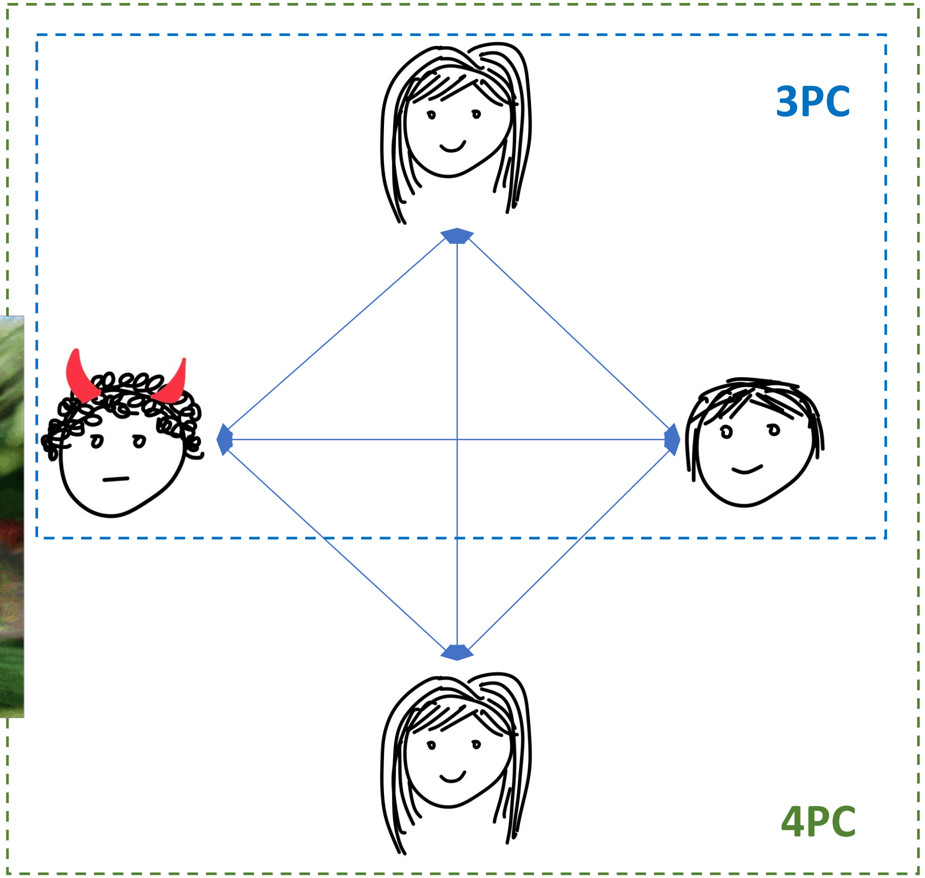
- 3PC and 4PC protocols for malicious corruption
- Honest-majority setting
  - at most 1 corruption
- Guaranteed Output Delivery
- Preprocessing Model
  - preprocessing phase



- ❑ input-independent computation
- ❑ relatively slow and expensive

SWIFT Protocol

- 3PC and 4PC protocols for malicious corruption
- Honest-majority setting
  - at most 1 corruption
- Guaranteed Output Delive
- Preprocessing Model
  - preprocessing phase
  - online phase



- ❑ input-dependent computation
- ❑ super fast

SWIFT Protocol: Results

- Communication cost per multiplication gate

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	Boyle et al. 19	-	3	GOD
	BLAZE [PS20]	3	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**

SWIFT Protocol: Results

- Communication cost per multiplication gate

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	Boyle et al. 19	-	3	GOD
	BLAZE [PS20]	3	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**

SWIFT Protocol: Results

- Communication cost per multiplication gate

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	Boyle et al. 19	-	3	GOD
	BLAZE [PS20]	3	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**

SWIFT Protocol: Results

- Communication cost per multiplication gate

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	Boyle et al. 19	-	3	GOD
	BLAZE [PS20]	3	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**
4PC	Trident [CRS20]	3	3	Fair
	FLASH [BCPS20]	6	6	GOD
	SWIFT (4PC)	**3**	**3**	**GOD**

SWIFT Protocol: Results

- Communication cost per multiplication gate

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	Boyle et al. 19	-	3	GOD
	BLAZE [PS20]	3	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**
4PC	Trident [CRS20]	3	3	Fair
	FLASH [BCPS20]	6	6	GOD
	SWIFT (4PC)	**3**	**3**	**GOD**

SWIFT Protocol: Results

- Communication cost per multiplication gate

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	Boyle et al. 19	-	3	GOD
	BLAZE [PS20]	3	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**
4PC	Trident [CRS20]	3	3	Fair
	FLASH [BCPS20]	6	6	GOD
	SWIFT (4PC)	**3**	**3**	**GOD**

SWIFT Protocol: Results in PPML

- PPML Building Blocks
  - Dot Product

$$\boxed{\vec{x} \odot \vec{y} = \sum_{i=1}^{d} x_i y_i}$$

SWIFT Protocol: Results in PPML

- PPML Building Blocks
  - Dot Product

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	BLAZE [PS20]	3d	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**
4PC	Trident [CRS20]	3	3	Fair
	FLASH [BCPS20]	6	6	GOD
	SWIFT (4PC)	**3**	**3**	**GOD**

Communication cost per dot product (d-length vectors)

$$\boxed{\vec{x} \odot \vec{y} = \sum_{i=1}^{d} x_i y_i}$$

SWIFT Protocol: Results in PPML

- PPML Building Blocks
  - Dot Product

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	BLAZE [PS20]	3d	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**
4PC	Trident [CRS20]	3	3	Fair
	FLASH [BCPS20]	6	6	GOD
	SWIFT (4PC)	**3**	**3**	**GOD**

Communication cost per dot product (d-length vectors)

$$\boxed{\vec{x} \odot \vec{y} = \sum_{i=1}^{d} x_i y_i}$$

SWIFT Protocol: Results in PPML

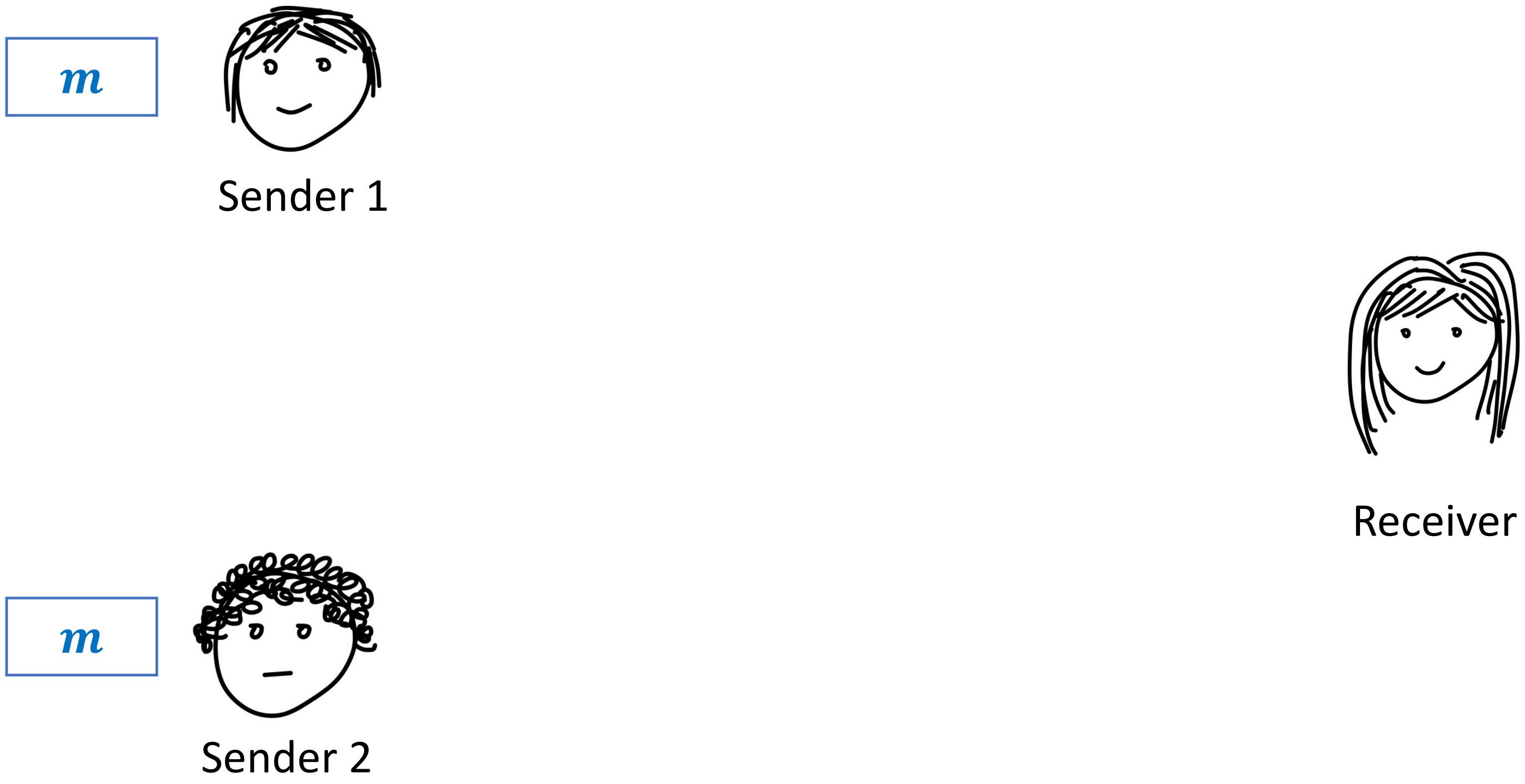
- PPML Building Blocks
  - Dot Product
  - Truncation
  - Comparison
  - Bit to arithmetic conversions
  - Non-linear activation functions

Setting	Reference	Pre-processing (#elements)	Online (#elements)	Security
3PC	BLAZE [PS20]	3d	3	Fair
	SWIFT (3PC)	**3**	**3**	**GOD**
4PC	Trident [CRS20]	3	3	Fair
	FLASH [BCPS20]	6	6	GOD
	SWIFT (4PC)	**3**	**3**	**GOD**

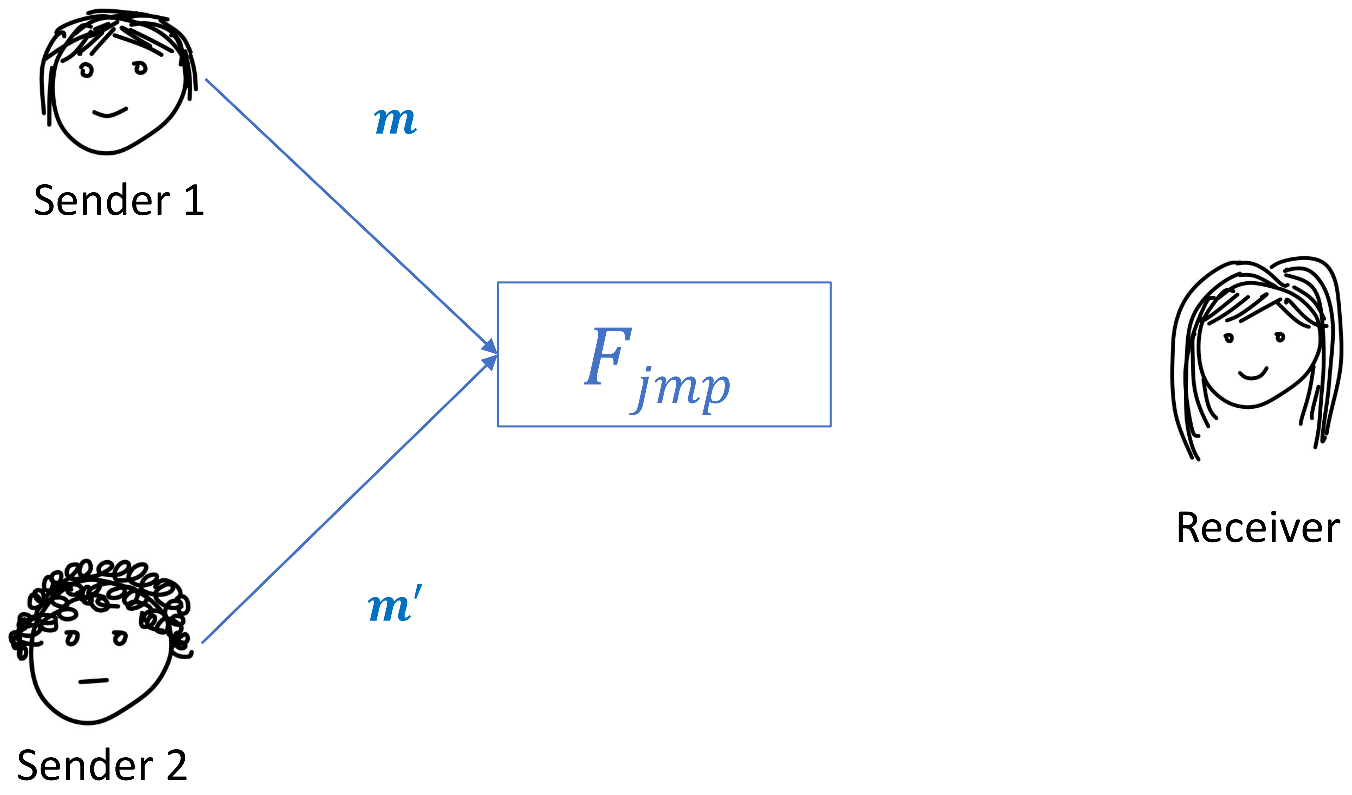
Communication cost per dot product (d-length vectors)

$$\boxed{\vec{x} \odot \vec{y} = \sum_{i=1}^{d} x_i y_i}$$

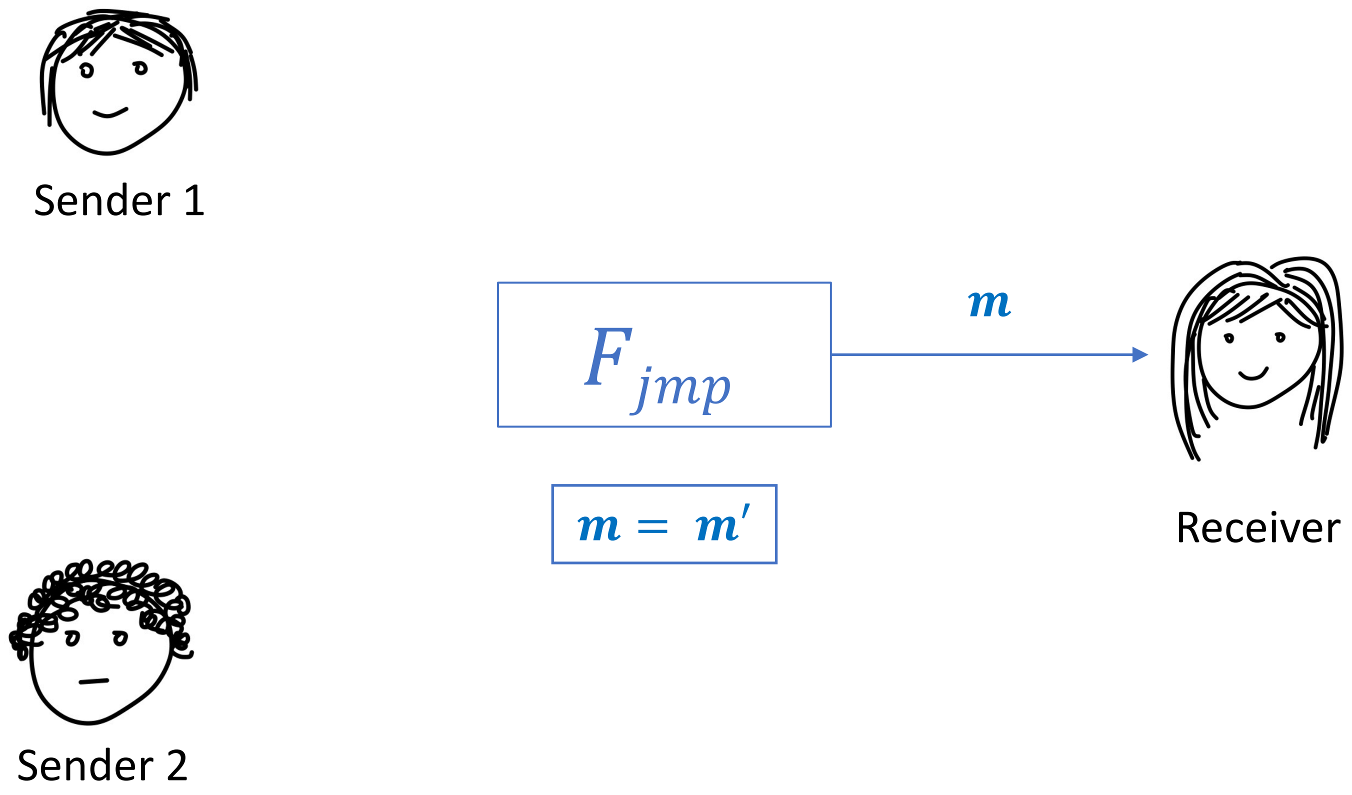
SWIFT Joint Message Passing (jmp) primitive



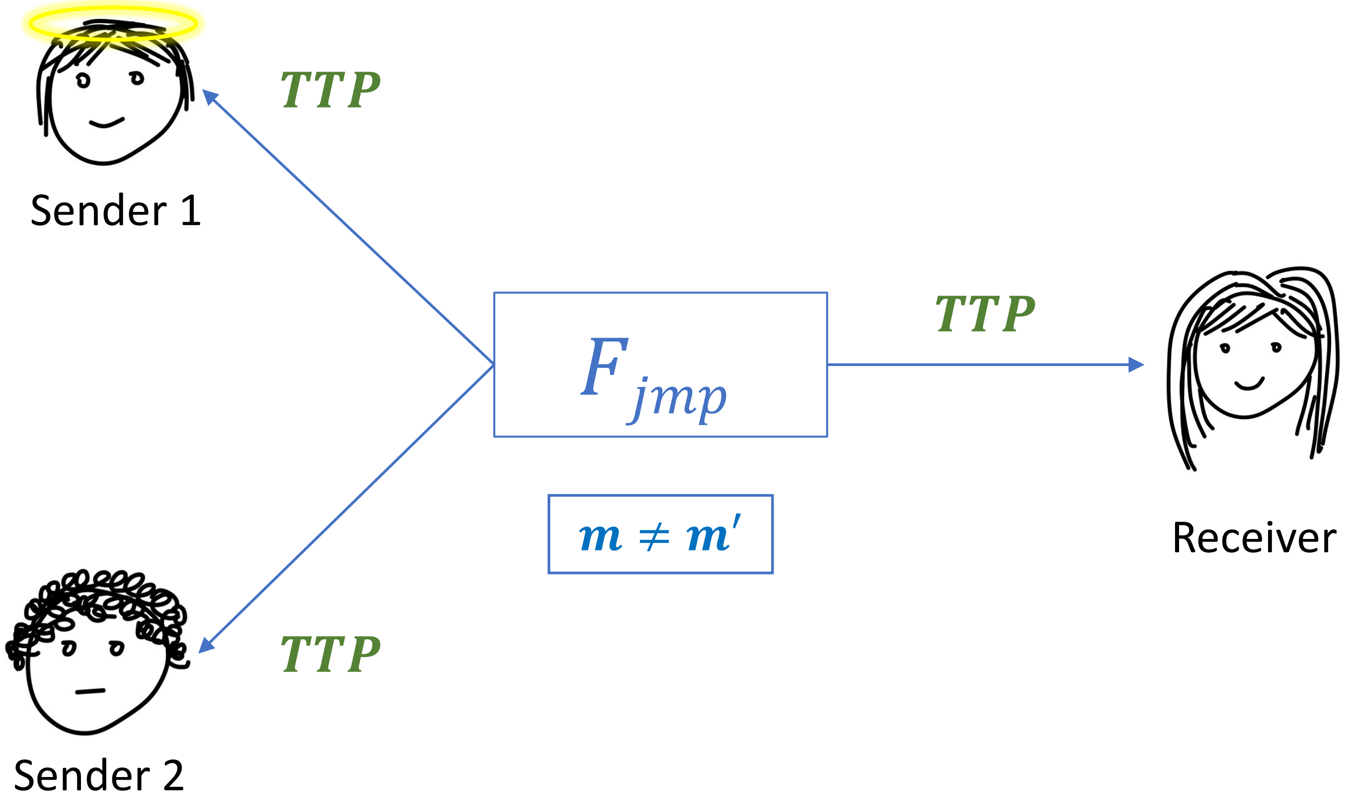
SWIFT Joint Message Passing (jmp) primitive



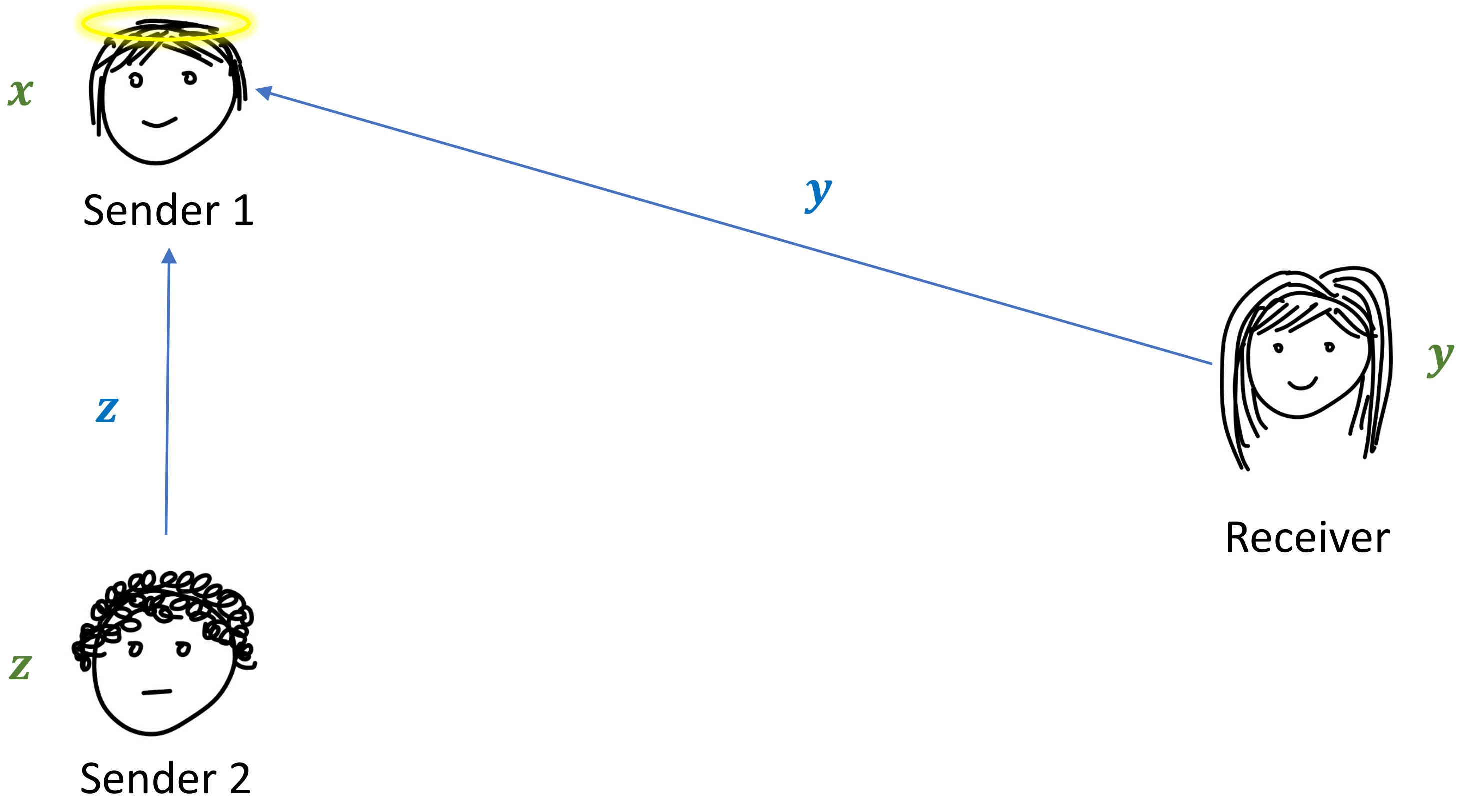
SWIFT Joint Message Passing (jmp) primitive



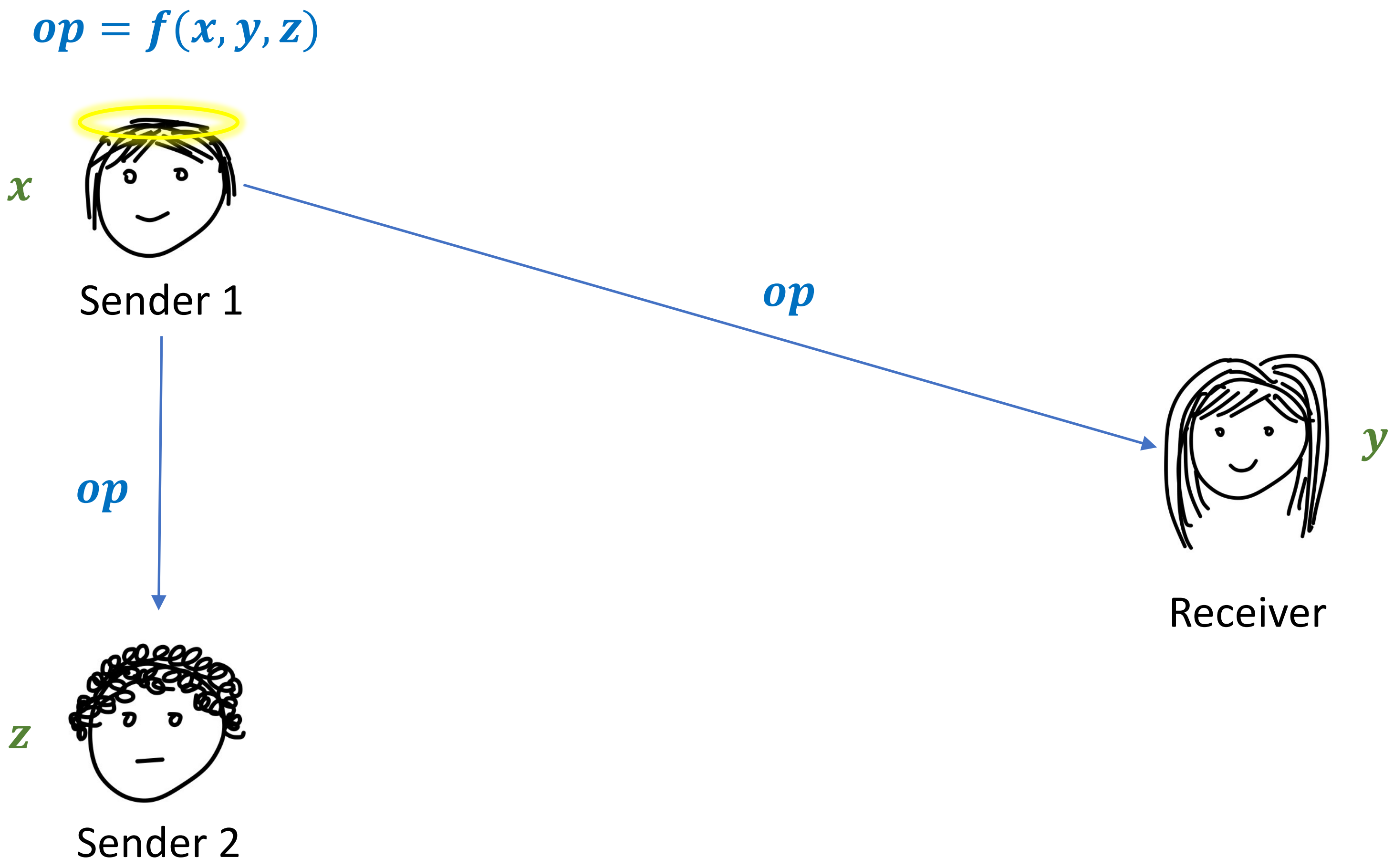
SWIFT Joint Message Passing (jmp) primitive



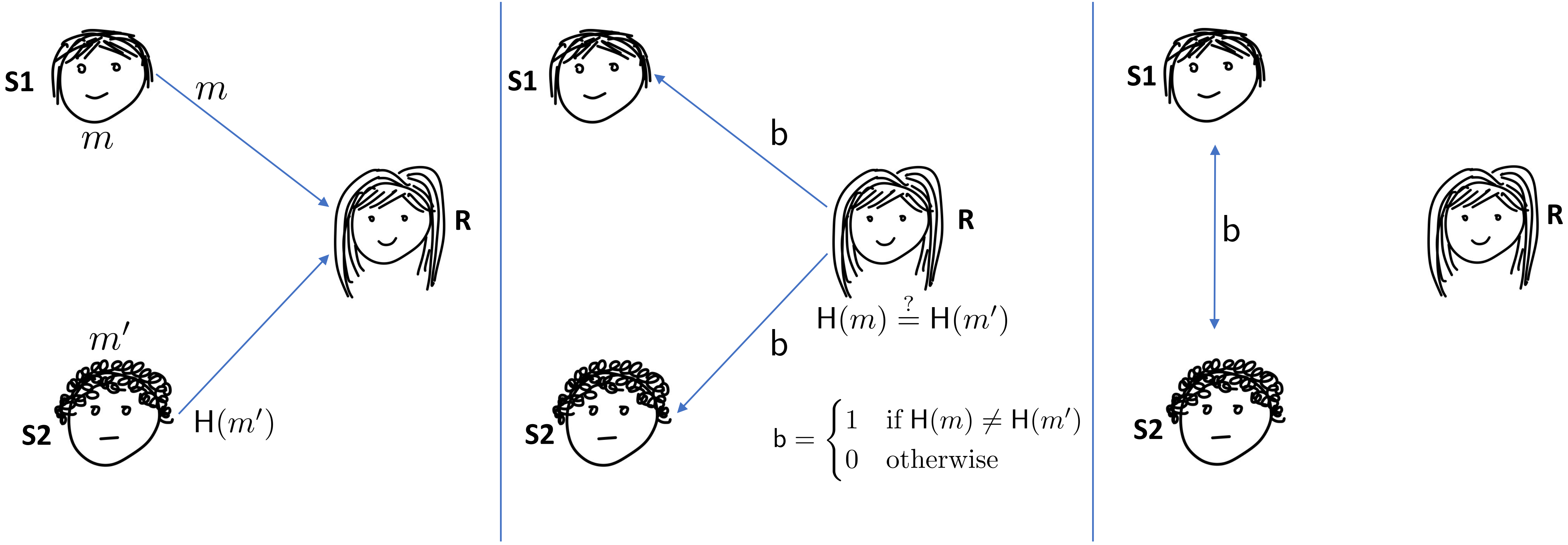
SWIFT Joint Message Passing (jmp) primitive



SWIFT Joint Message Passing (jmp) primitive



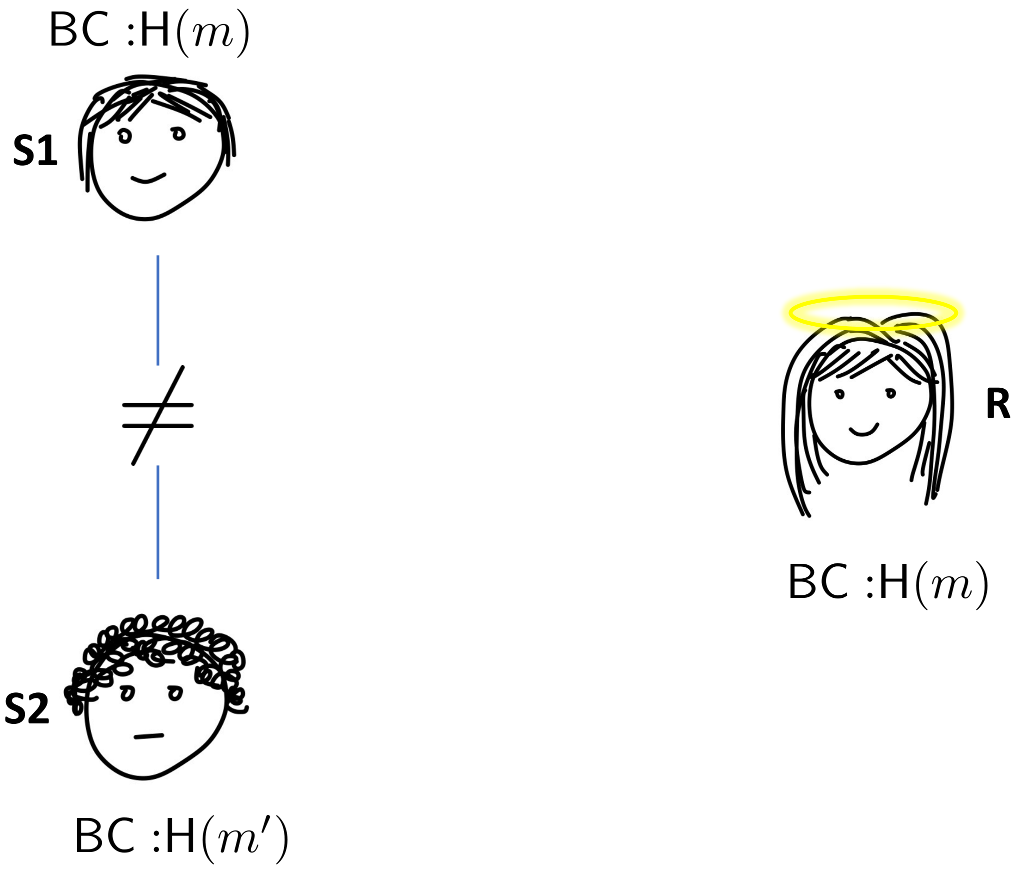
3PC Joint Message Passing (jmp) primitive



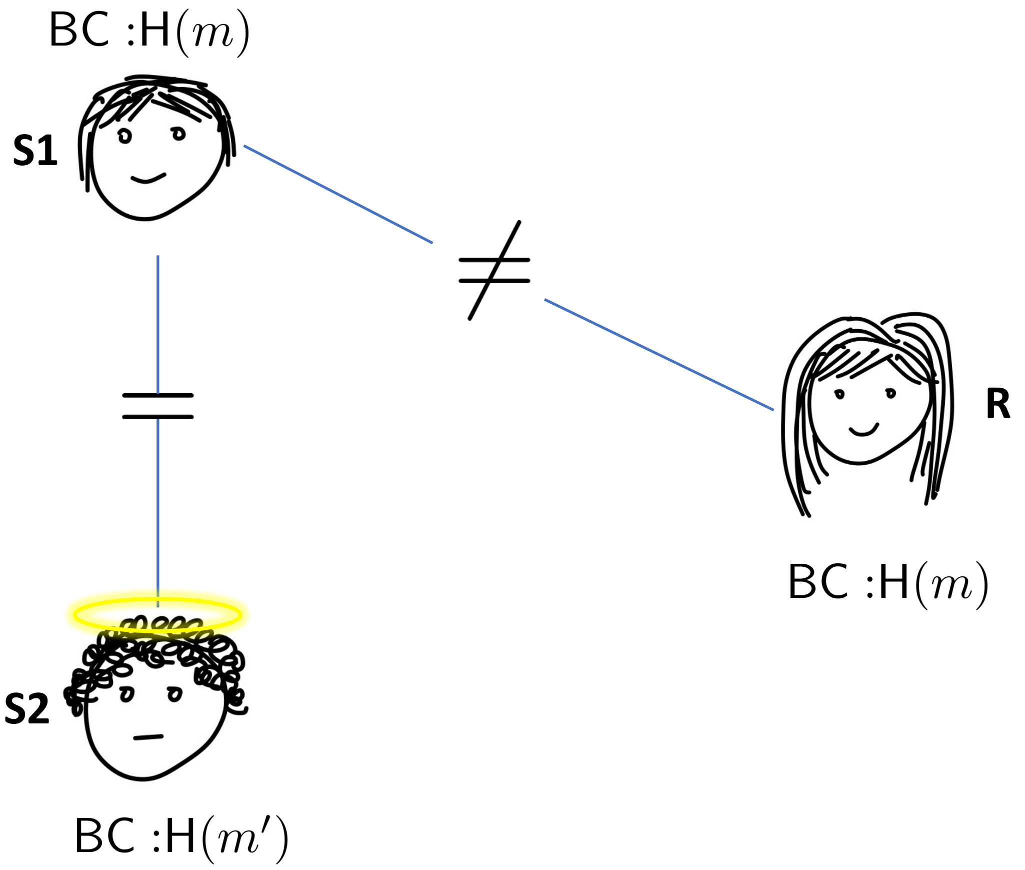
3PC Joint Message Passing (jmp) primitive

$$b = \begin{cases} 1 & \text{if } \mathsf{H}(m) \neq \mathsf{H}(m') \\ 0 & \text{otherwise} \end{cases}$$

If b = 1

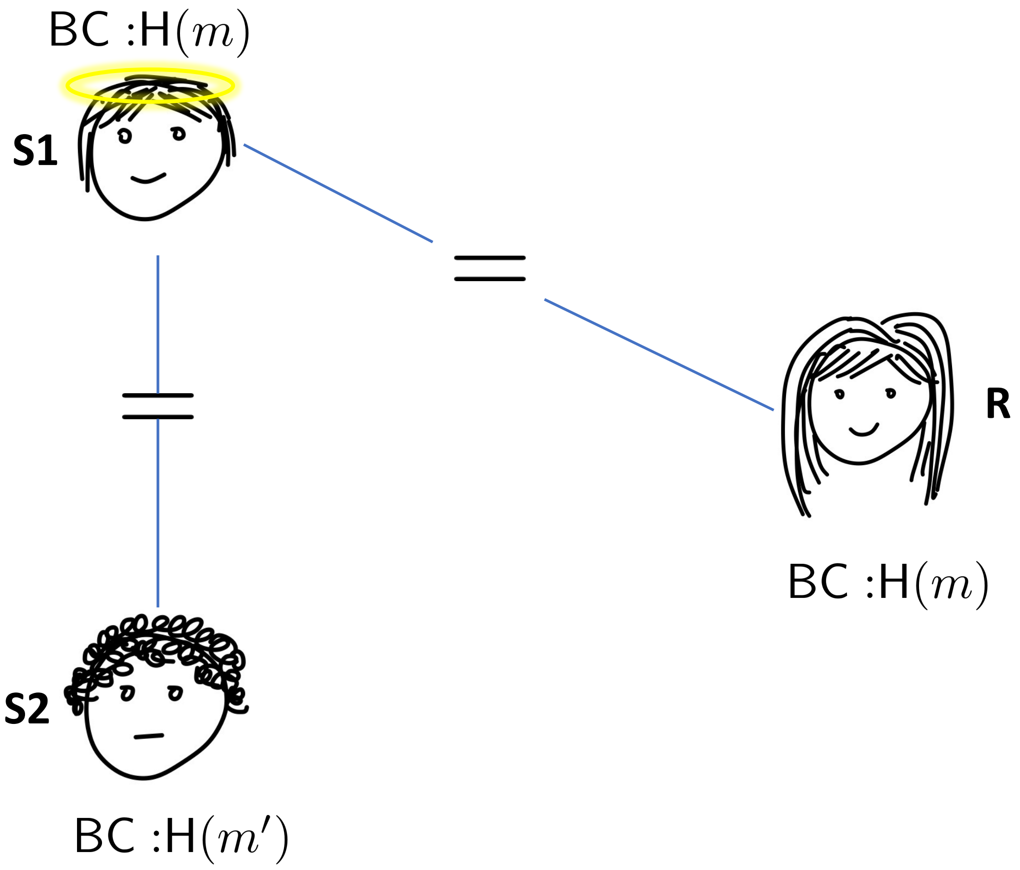


3PC Joint Message Passing (jmp) primitive



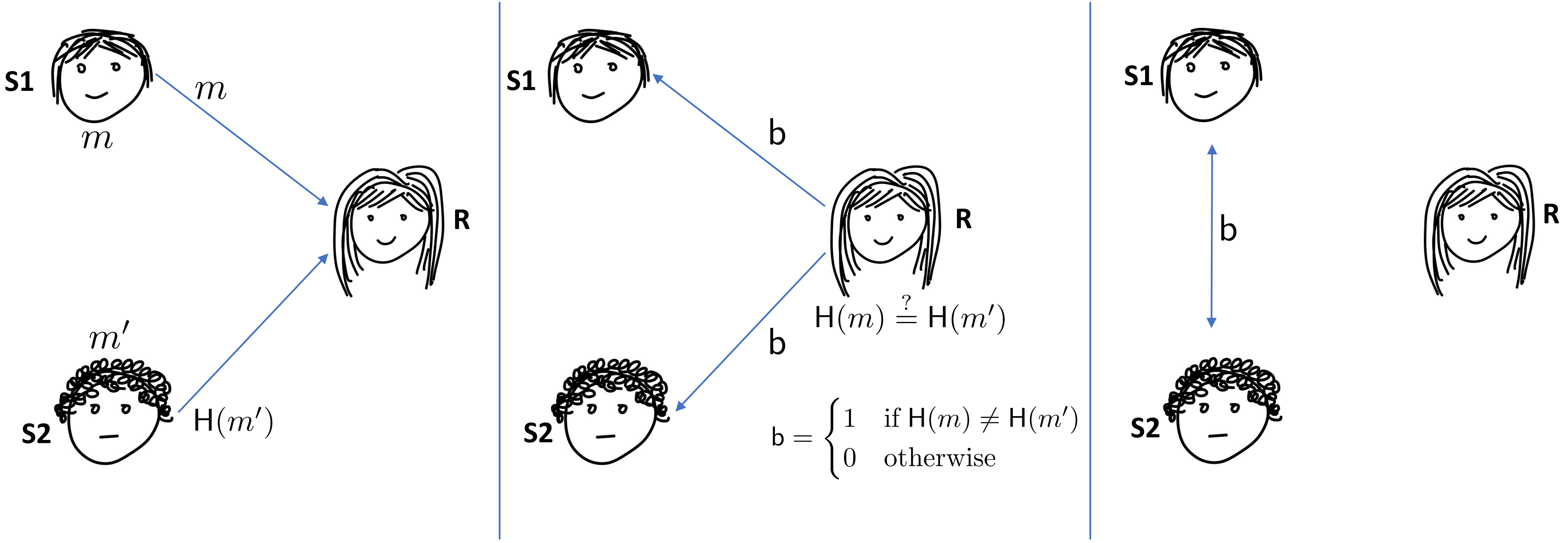
$$b = \begin{cases} 1 & \text{if } \mathsf{H}(m) \neq \mathsf{H}(m') \\ 0 & \text{otherwise} \end{cases}$$

3PC Joint Message Passing (jmp) primitive

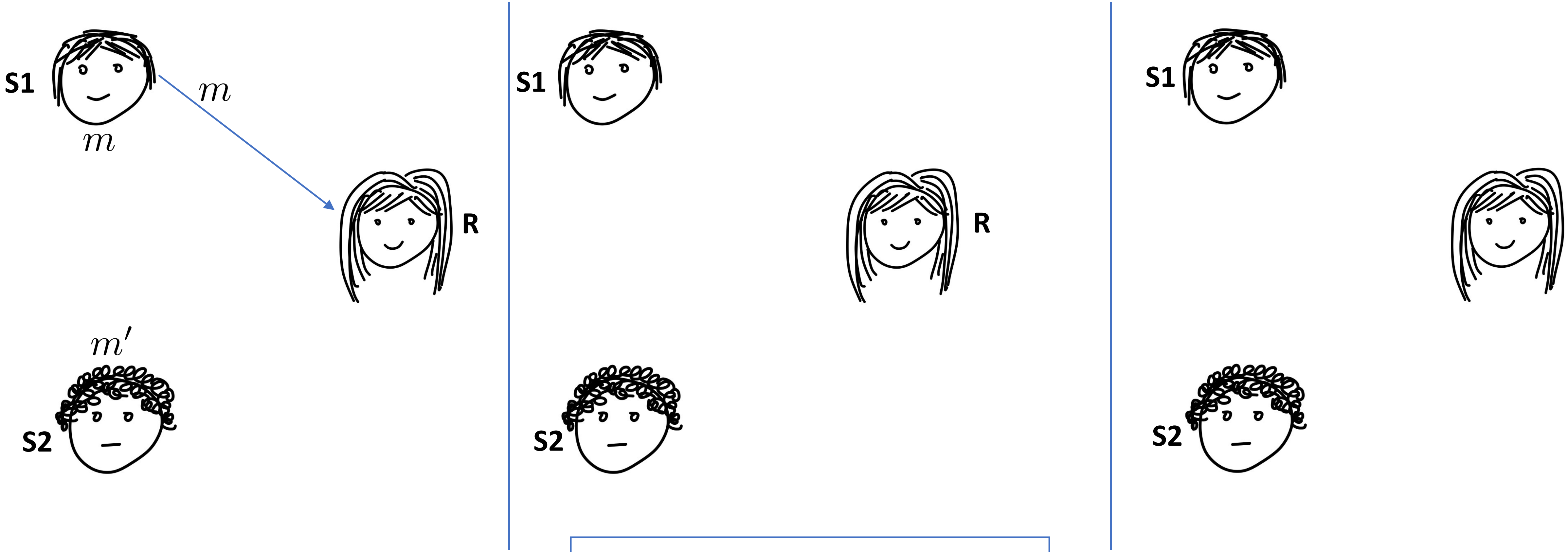


$$\mathsf{b} = \begin{cases} 1 & \text{if } \mathsf{H}(m) \neq \mathsf{H}(m') \\ 0 & \text{otherwise} \end{cases}$$

3PC Joint Message Passing (jmp) primitive

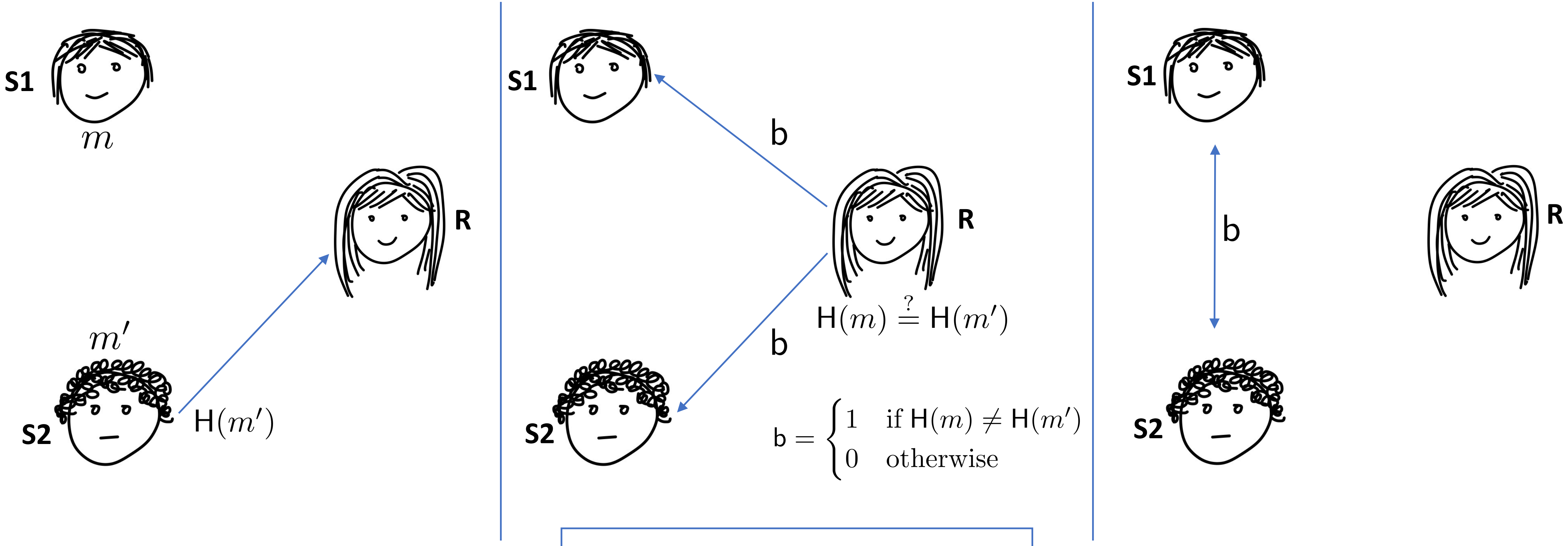


3PC Joint Message Passing (jmp) primitive



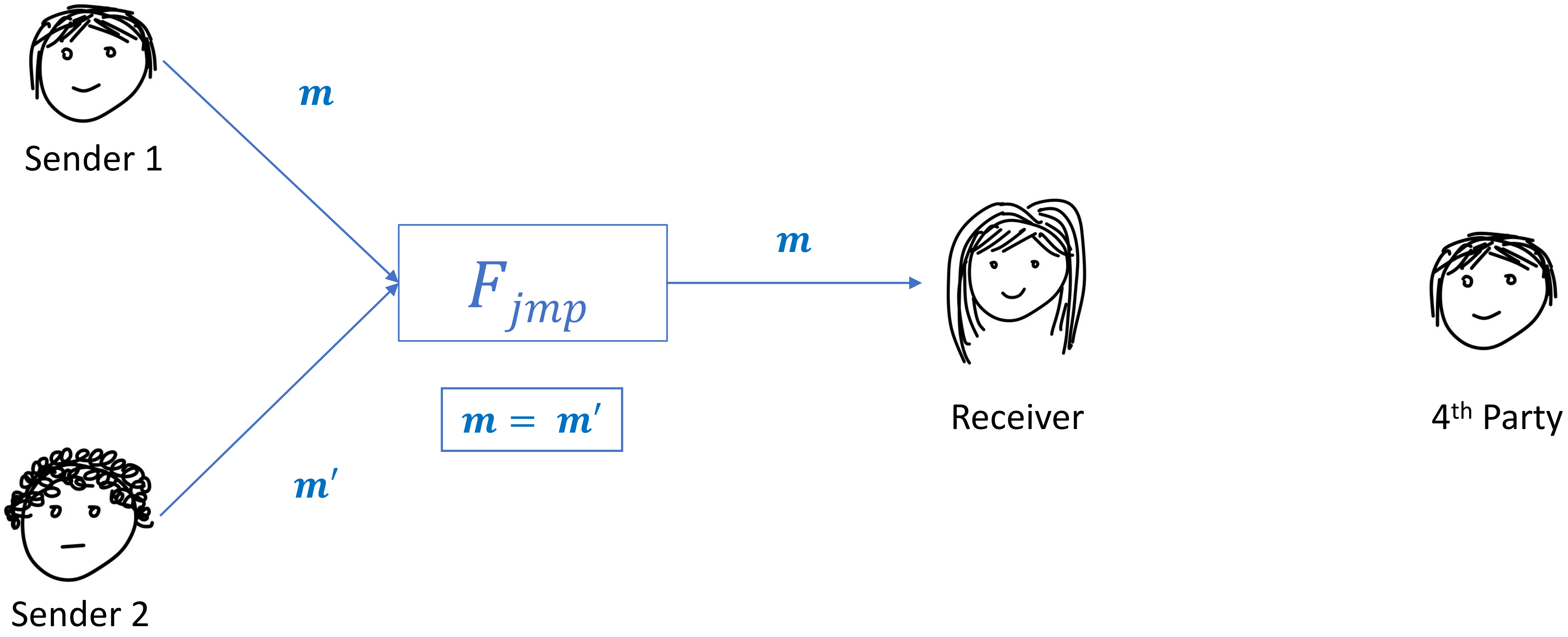
Send phase of jmp

3PC Joint Message Passing (jmp) primitive



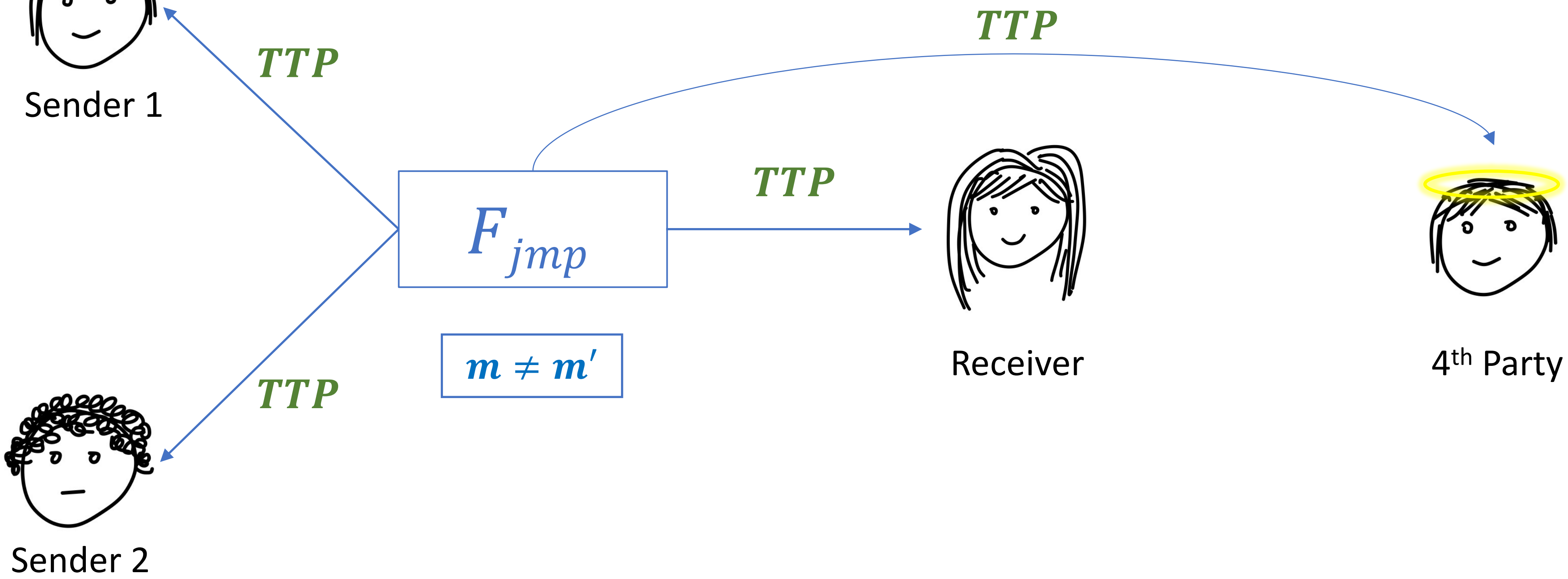
Verify phase of jmp

4PC Joint Message Passing (jmp) primitive



jmp for 4PC

4PC Joint Message Passing (jmp) primitive



jmp for 4PC

SWIFT: Benchmarking

Setting	Ref.	Online (TP in $\times 10^3$)			Total	
		Latency (s)	Com [KB]	TP	Latency (s)	Com [KB]
3PC Training	BLAZE	0.74	50.26	4872.38	0.93	203.35
	SWIFT	1.05	50.32	4872.38	1.54	203.47
3PC Inference	BLAZE	0.66	0.28	7852.05	0.84	0.74
	SWIFT	0.97	0.34	6076.46	1.46	0.86
4PC Training	FLASH	0.83	88.93	5194.18	1.11	166.75
	SWIFT	0.83	41.32	11969.48	1.11	92.91
4PC Inference	FLASH	0.76	0.50	7678.40	1.04	0.96
	SWIFT	0.75	0.27	15586.96	1.03	0.57

Table 6: Logistic Regression training and inference. TP is given in (#it/min) for training and (#queries/min) for inference.

SWIFT: Benchmarking

Network	Ref.	Online			Total	
		Latency (s)	Com [MB]	TP	Latency (s)	Com [MB]
NN-1	BLAZE	1.92	0.04	49275.19	2.35	0.11
	SWIFT	2.22	0.04	49275.19	2.97	0.11
NN-2	BLAZE	4.77	3.54	536.52	5.61	9.59
	SWIFT	5.08	3.54	536.52	6.22	9.59
NN-3	BLAZE	15.58	52.58	36.03	18.81	148.02
	SWIFT	15.89	52.58	36.03	19.29	148.02

Table 7: 3PC NN Inference. TP is given in (#queries/min).

NN-1: [MR18, PS20] NN-2: [LBBH98] NN-3: [SZ14]

Network	Ref.	Online			Total	
		Latency (s)	Com [MB]	TP	Latency (s)	Com [MB]
NN-1	FLASH	1.70	0.06	59130.23	2.17	0.12
	SWIFT	1.70	0.03	147825.56	2.17	0.06
NN-2	FLASH	3.93	5.51	653.67	4.71	10.50
	SWIFT	3.93	2.33	1672.55	4.71	5.40
NN-3	FLASH	12.65	82.54	43.61	15.31	157.11
	SWIFT	12.50	35.21	110.47	15.14	81.46

Table 8: 4PC NN Inference. TP is given in (#queries/min).

Thank you