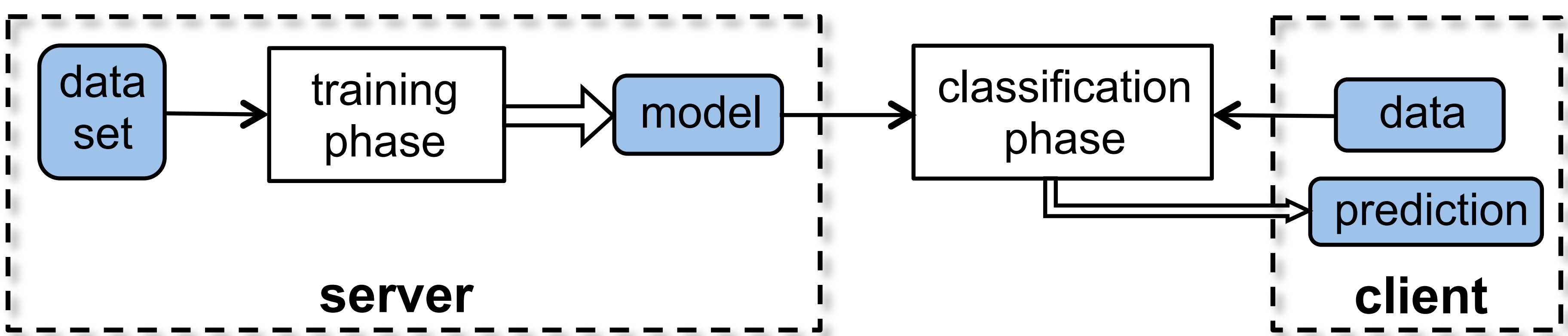
Machine Learning Classification over Encrypted Data

Raphael Bost, Raluca Ada Popa,
Stephen Tu, Shafi Goldwasser

Classification
(Machine Learning)

- Supervised learning (training)
- Classification



Secure Classification

- The provider's model is sensible

  financial model, genetic sequences, …

- Client's private data

  medical records, credit history, …

Secure Classification

- The provider’s model is sensible

  financial model, genetic sequences, …

- Client’s private data

  medical records, credit history, …

MPC / 2PC

Using General 2PC ?

+ Works for every circuit

+ Constant number of interactions

- Have to build circuits

- Hard to ‘compose’

- Not easily reusable

➡ *Ad Hoc* protocols

Scope of our work

- Secure classification, no learning
  the model is already known
- Differential privacy is out of scope
  can be treated separately
- Classifiers as specialized 2PC, but not a specialized classifier

Approach

- Security model: passive (honest-but-curious) adversary
- Identify and construct reusable building blocks
- Practical performance as a primary goal
- Choose the best fitted primitives

  Homomorphic Encryption, FHE, Garbled Circuits, …

Building Blocks

- Dot product
- Encrypted Comparison
- Encrypted (arg)max
- Decision trees
- Encryption scheme switching

Argmax

- Alice $(\llbracket a_1 \rrbracket, \ldots, \llbracket a_n \rrbracket, \mathsf{PK})$
- Bob SK
- The comparison pattern must not depend on the values

Argmax

- Alice $(\llbracket a_1 \rrbracket, \ldots, \llbracket a_n \rrbracket, \mathsf{PK})$
- Bob SK
- The comparison pattern must not depend on the values
    - Compare everything

Argmax

- Alice $(\llbracket a_1 \rrbracket, \ldots, \llbracket a_n \rrbracket, \mathsf{PK})$
- Bob SK
- The comparison pattern must not depend on the values
  - Compare everything $\Rightarrow O(n^2)$

Argmax

- Alice $(\llbracket a_1 \rrbracket, \ldots, \llbracket a_n \rrbracket, \mathsf{PK})$
- Bob SK
- The comparison pattern must not depend on the values
    - ~~Compare everything~~ $\Rightarrow O(n^2)$

Argmax

- Alice $(\llbracket a_1 \rrbracket, \ldots, \llbracket a_n \rrbracket, \mathsf{PK})$
- Bob SK
- The comparison pattern must not depend on the values
  - ~~Compare everything~~ $\Rightarrow O(n^2)$
  - 'Classical' algorithm

Argmax

- Alice $([\![a_1]\!], \ldots, [\![a_n]\!], \mathsf{PK})$

- Bob SK

- The comparison pattern must not depend on the values

  - ~~Compare everything~~ $\Rightarrow O(n^2)$

  - 'Classical' algorithm $\Rightarrow O(n)$

Compare & Swap

Alice

$(\mathsf{PK}, [\![v]\!], [\![w]\!])$

$[\![\max(v, w)]\!]$

Bob SK

$(v < w)$

Compare & Swap

Alice

$(\mathsf{PK}, [\![v]\!], [\![w]\!])$

$\varnothing$

$[\![\max(v, w)]\!]$

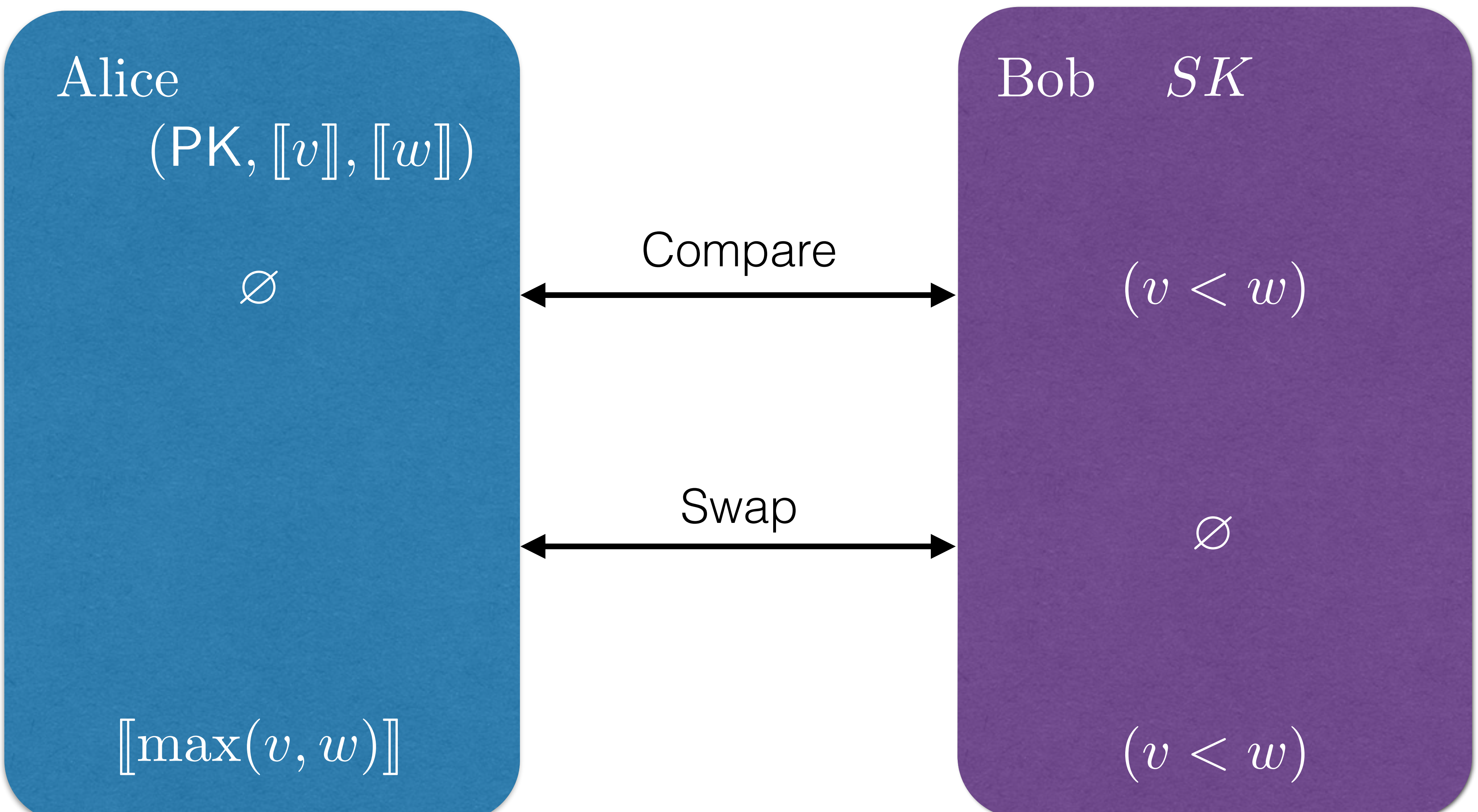
$\xleftrightarrow{\text{Compare}}$

Bob SK

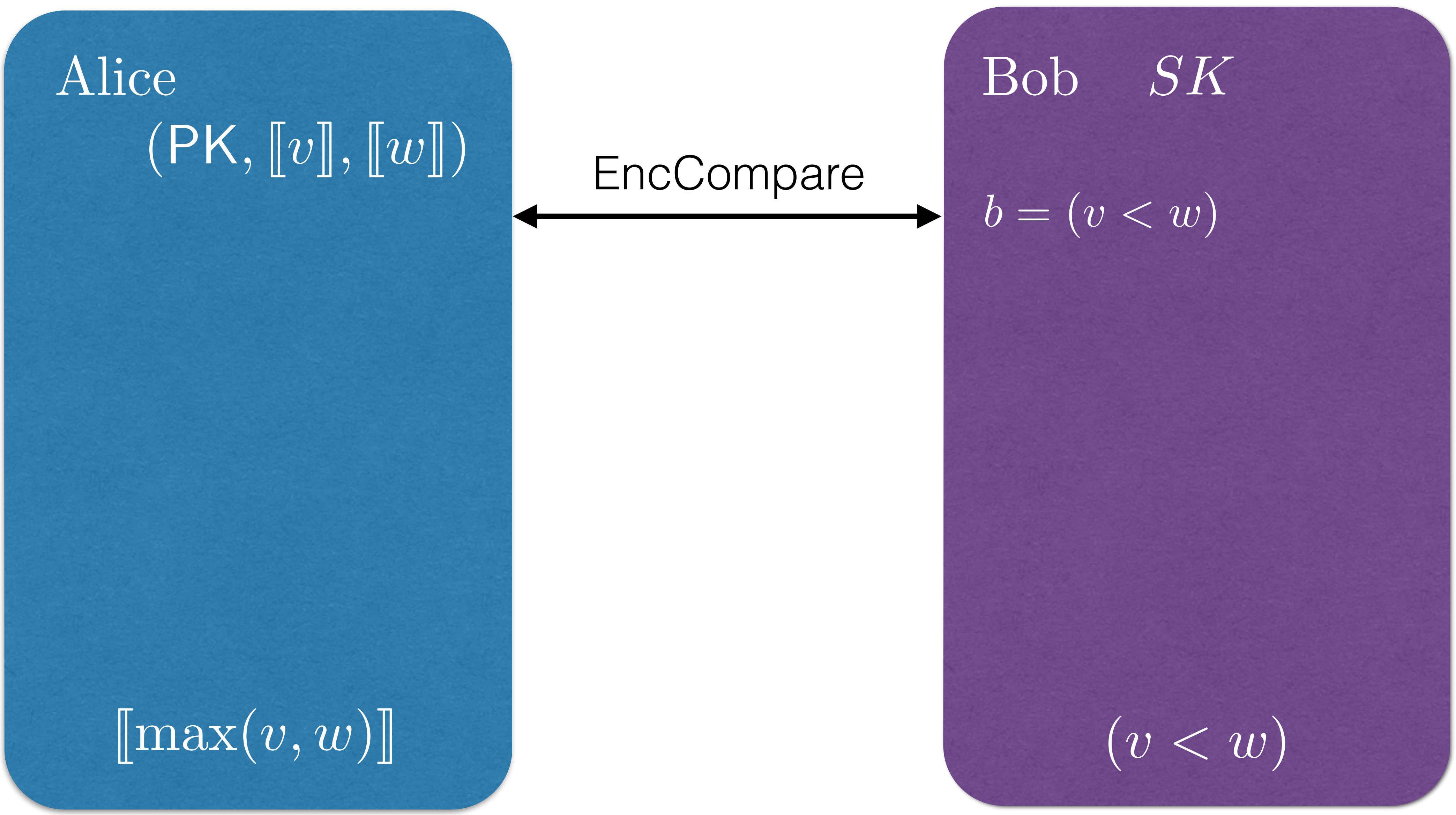
$(v < w)$

$(v < w)$

Compare & Swap



Compare & Swap



Compare & Swap

Alice

$(\mathsf{PK}, [\![v]\!], [\![w]\!])$

$(r, s) \leftarrow M^2$

$[\![v']\!] = [\![v + r]\!]$

$[\![w']\!] = [\![w + s]\!]$

$[\![\max(v, w)]\!]$

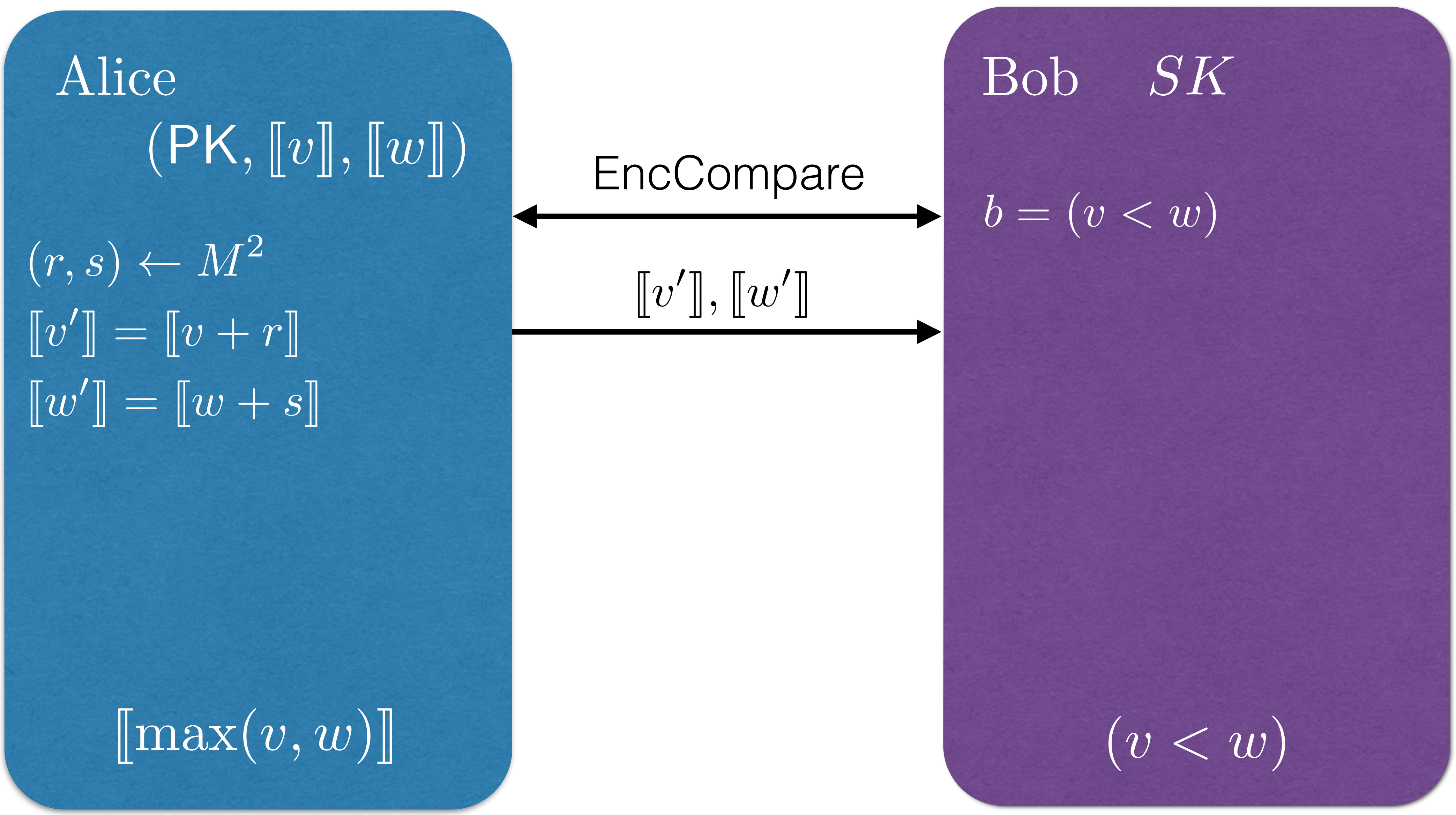
EncCompare

Bob SK

$b = (v < w)$

$(v < w)$

Compare & Swap



Compare & Swap

Alice

$(\mathsf{PK}, \llbracket v \rrbracket, \llbracket w \rrbracket)$

$(r, s) \leftarrow M^2$

$\llbracket v' \rrbracket = \llbracket v + r \rrbracket$

$\llbracket w' \rrbracket = \llbracket w + s \rrbracket$

$\llbracket \max(v, w) \rrbracket$

$\xleftrightarrow{\text{EncCompare}}$

$\xrightarrow{\llbracket v' \rrbracket, \llbracket w' \rrbracket}$

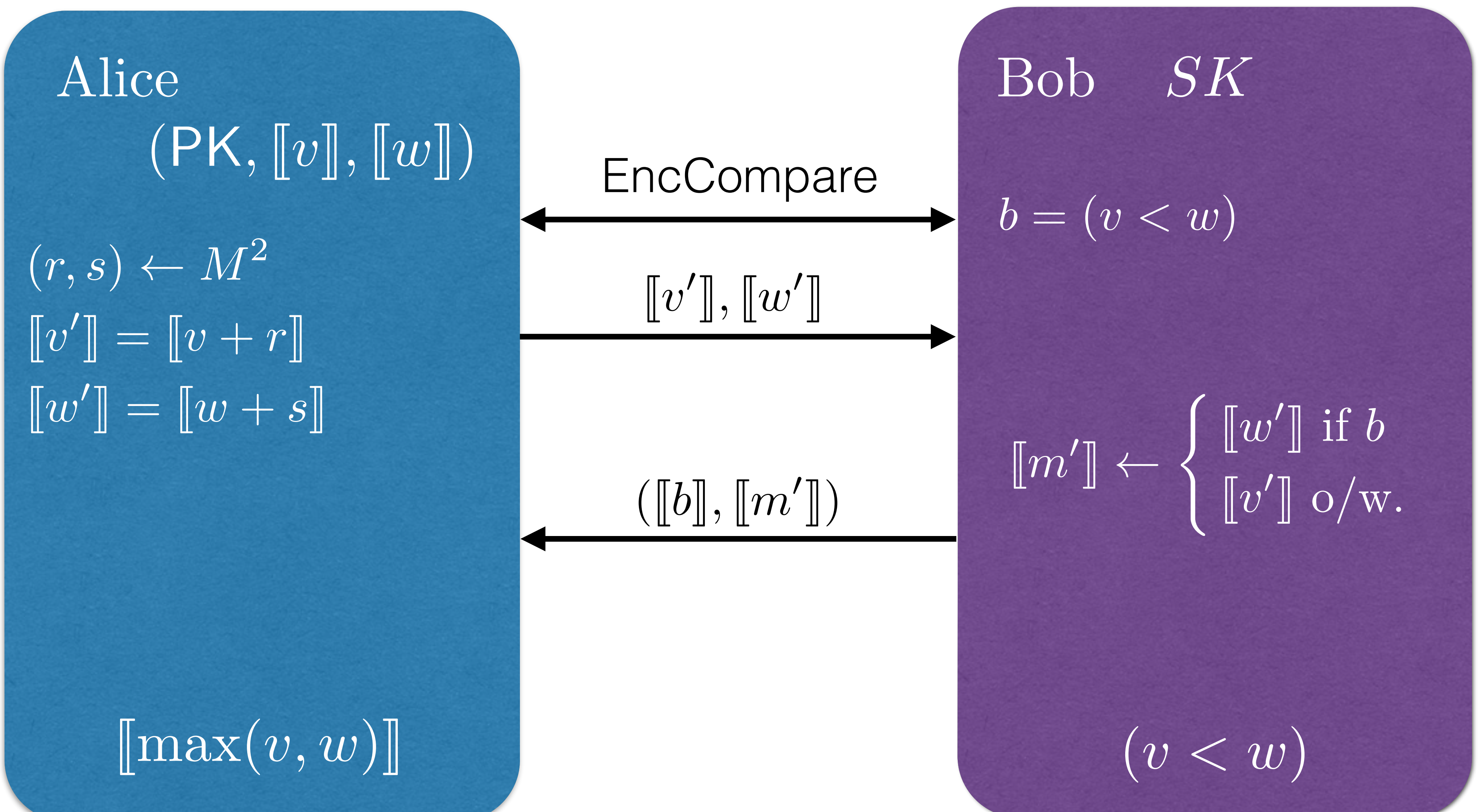
Bob SK

$b = (v < w)$

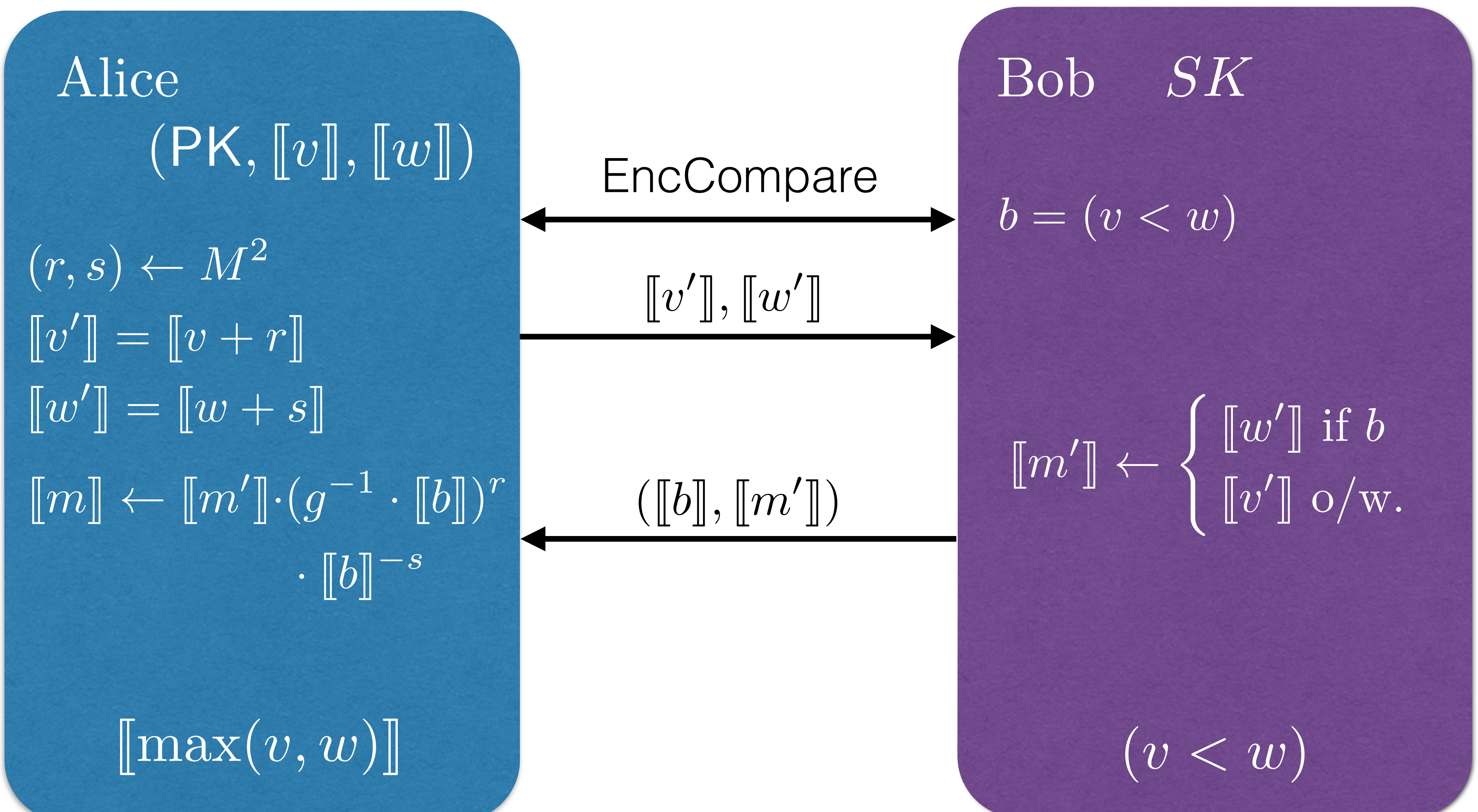
$$\llbracket m' \rrbracket \leftarrow \begin{cases} \llbracket w' \rrbracket & \text{if } b \\ \llbracket v' \rrbracket & \text{o/w.} \end{cases}$$

$(v < w)$

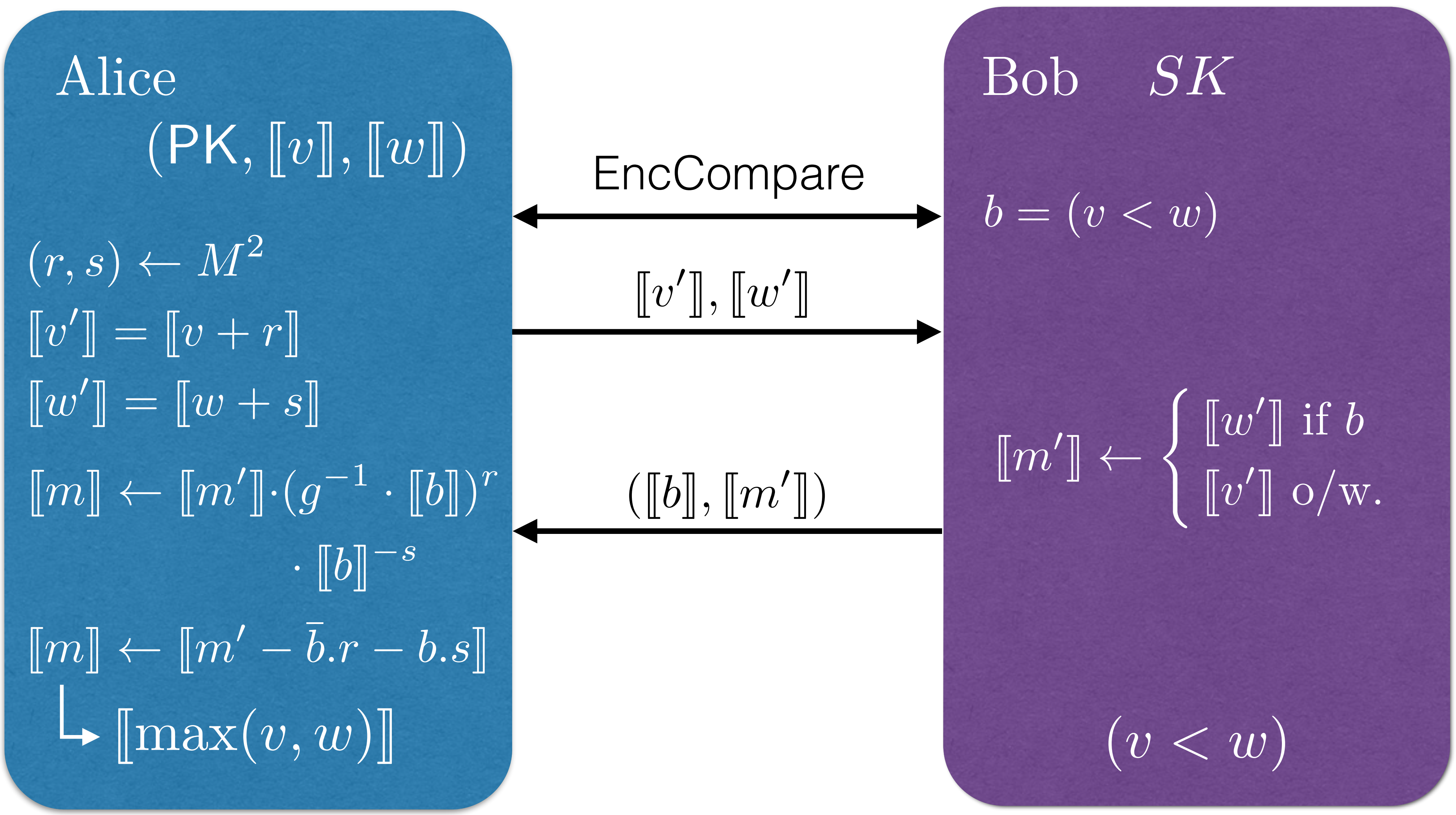
Compare & Swap



Compare & Swap



Compare & Swap



Argmax

- Protocol : n-1 Compare & Swap

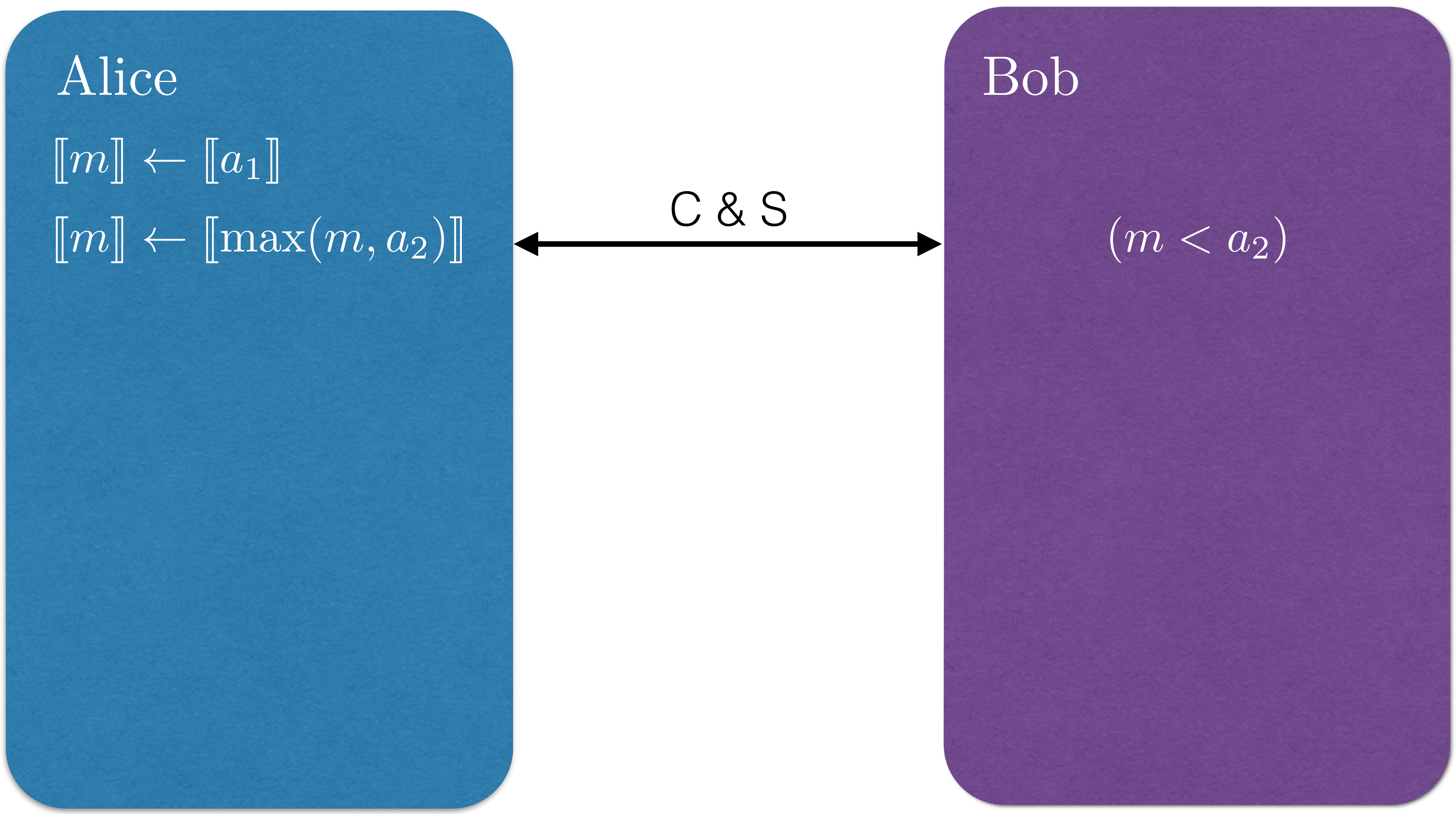
Alice

$[\![m]\!] \leftarrow [\![a_1]\!]$

Bob

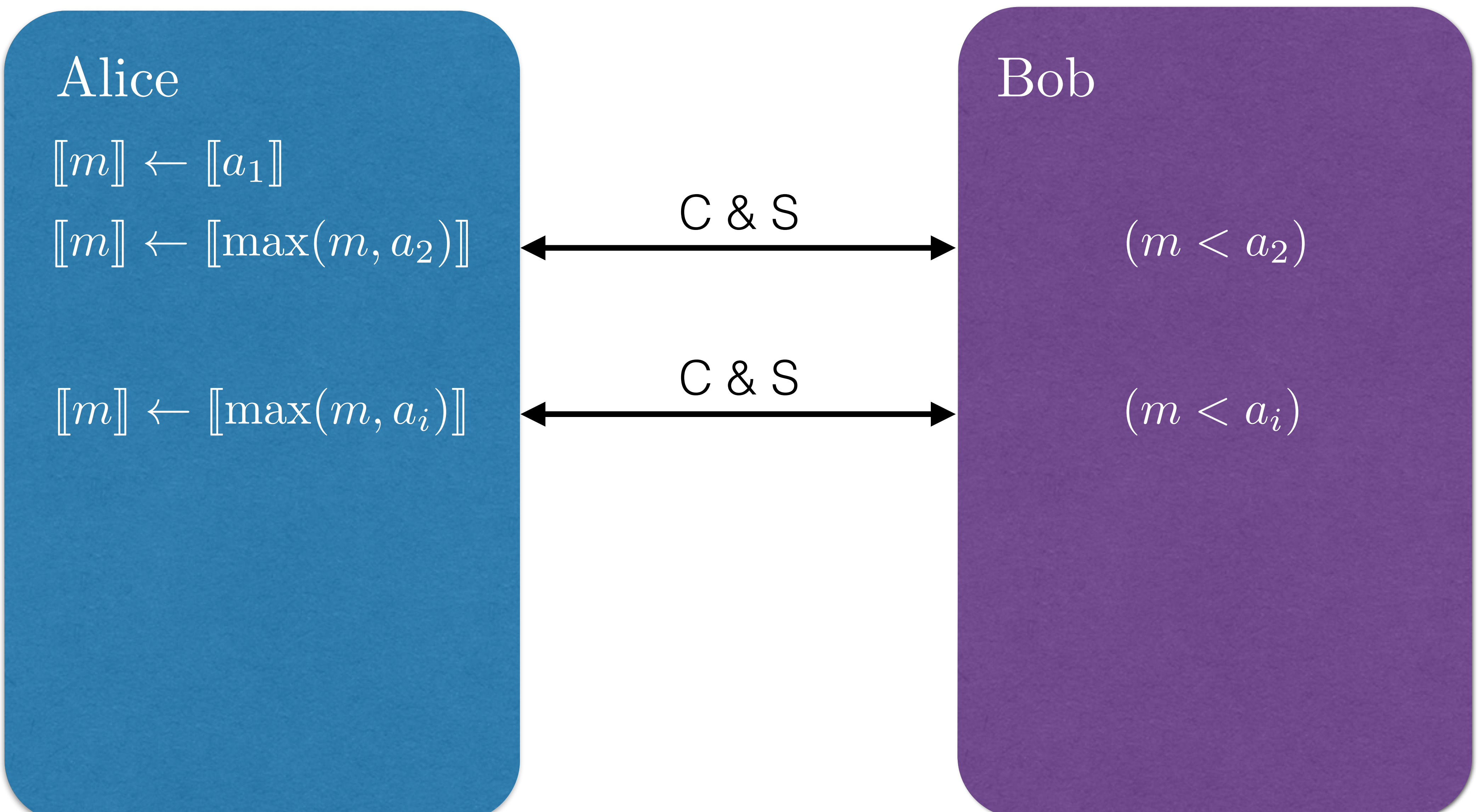
Argmax

- Protocol : n-1 Compare & Swap



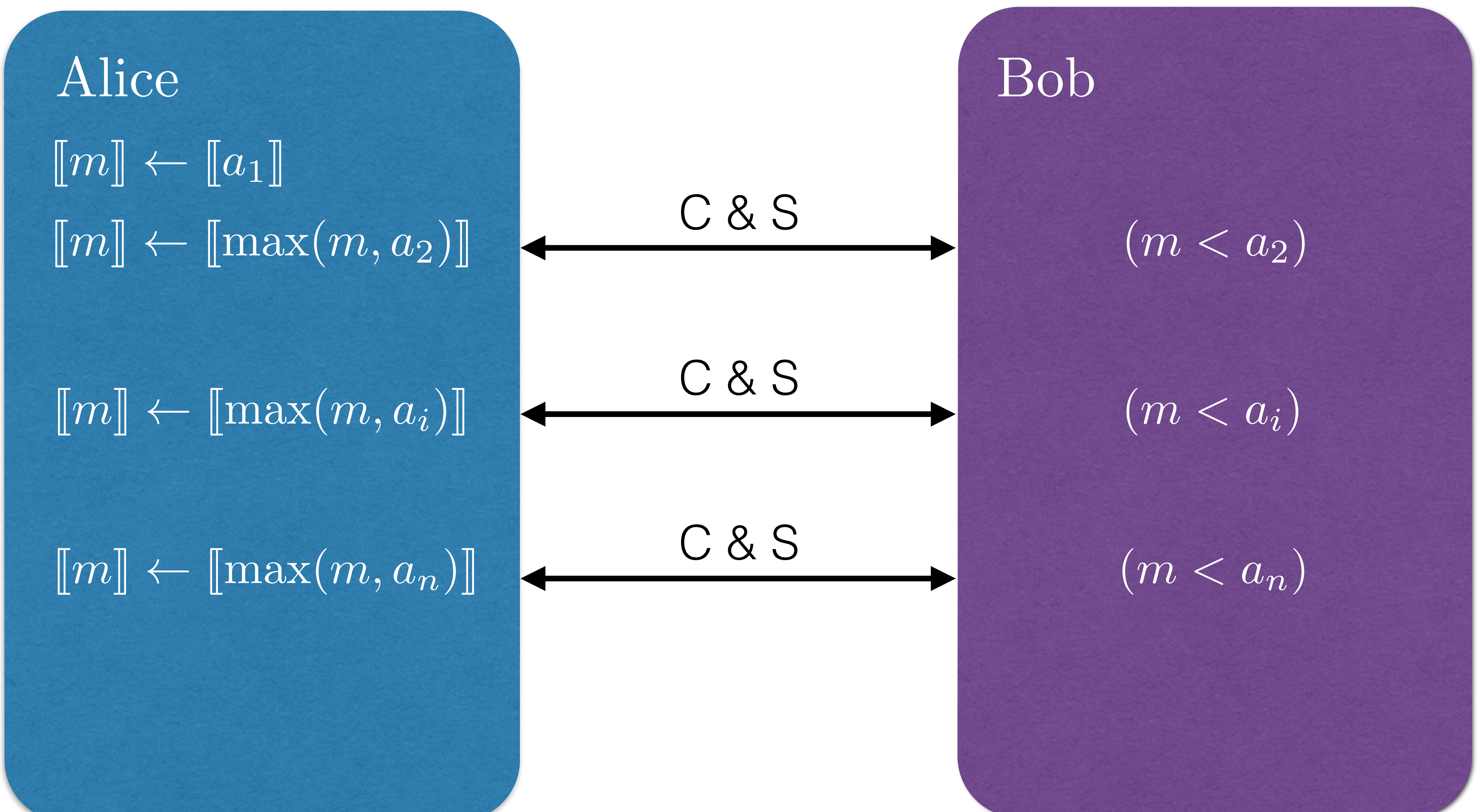
Argmax

- Protocol : n-1 Compare & Swap



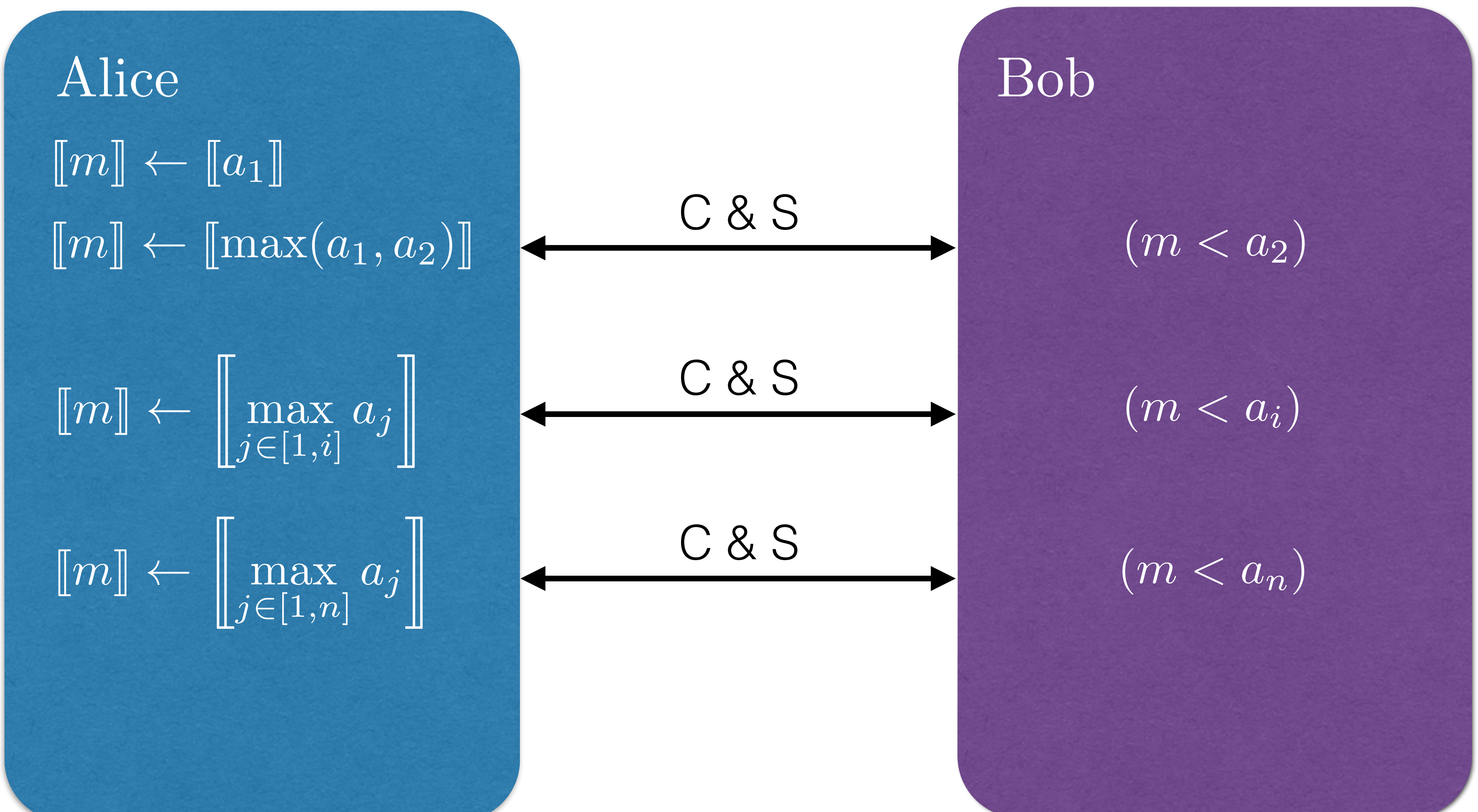
Argmax

- Protocol : n-1 Compare & Swap



Argmax

- Protocol : n-1 Compare & Swap



Argmax

- Protocol : n-1 Compare & Swap

Alice		Bob
$[\![m]\!] \leftarrow [\![a_1]\!]$		
$[\![m]\!] \leftarrow [\![\max(a_1, a_2)]\!]$	C & S $\longleftrightarrow$	$(a_1 < a_2)$
$[\![m]\!] \leftarrow \left[\!\!\left[\max\limits_{j \in [1,i]} a_j\right]\!\!\right]$	C & S $\longleftrightarrow$	$(m < a_i) \Rightarrow \operatorname*{argmax}\limits_{j \in [1,i]} a_j$
$[\![m]\!] \leftarrow \left[\!\!\left[\max\limits_{j \in [1,n]} a_j\right]\!\!\right]$	C & S $\longleftrightarrow$	$(m < a_n) \Rightarrow \operatorname*{argmax}\limits_{j \in [1,n]} a_j$

Argmax

- Protocol : n-1 Compare & Swap

Alice

$[\![m]\!] \leftarrow [\![a_{\pi(1)}]\!]$

$[\![m]\!] \leftarrow [\![\max(a_{\pi(1)}, a_{\pi(2)})]\!]$

$[\![m]\!] \leftarrow \left[\!\!\left[\max_{j\in[1,i]} a_{\pi(j)}\right]\!\!\right]$

$[\![m]\!] \leftarrow \left[\!\!\left[\max_{j\in[1,n]} a_{\pi(j)}\right]\!\!\right]$

$\boxed{\max a_j}$

C & S

C & S

C & S

Bob

$(a_{\pi(1)} < a_{\pi(1)})$

$(m < a_{\pi(i)})$

$\Rightarrow \operatorname*{argmax}_{j\in[1,i]} a_{\pi(j)}$

$(m < a_{\pi(n)})$

$\Rightarrow \operatorname*{argmax}_{j\in[1,n]} a_{\pi(j)}$

$\boxed{\pi(\operatorname{argmax} a_j)}$

Argmax

- Protocol : n-1 Compare & Swap

Argmax

- Protocol : n-1 Compare & Swap sequentially

Argmax

- Protocol : n-1 Compare & Swap

  sequentially

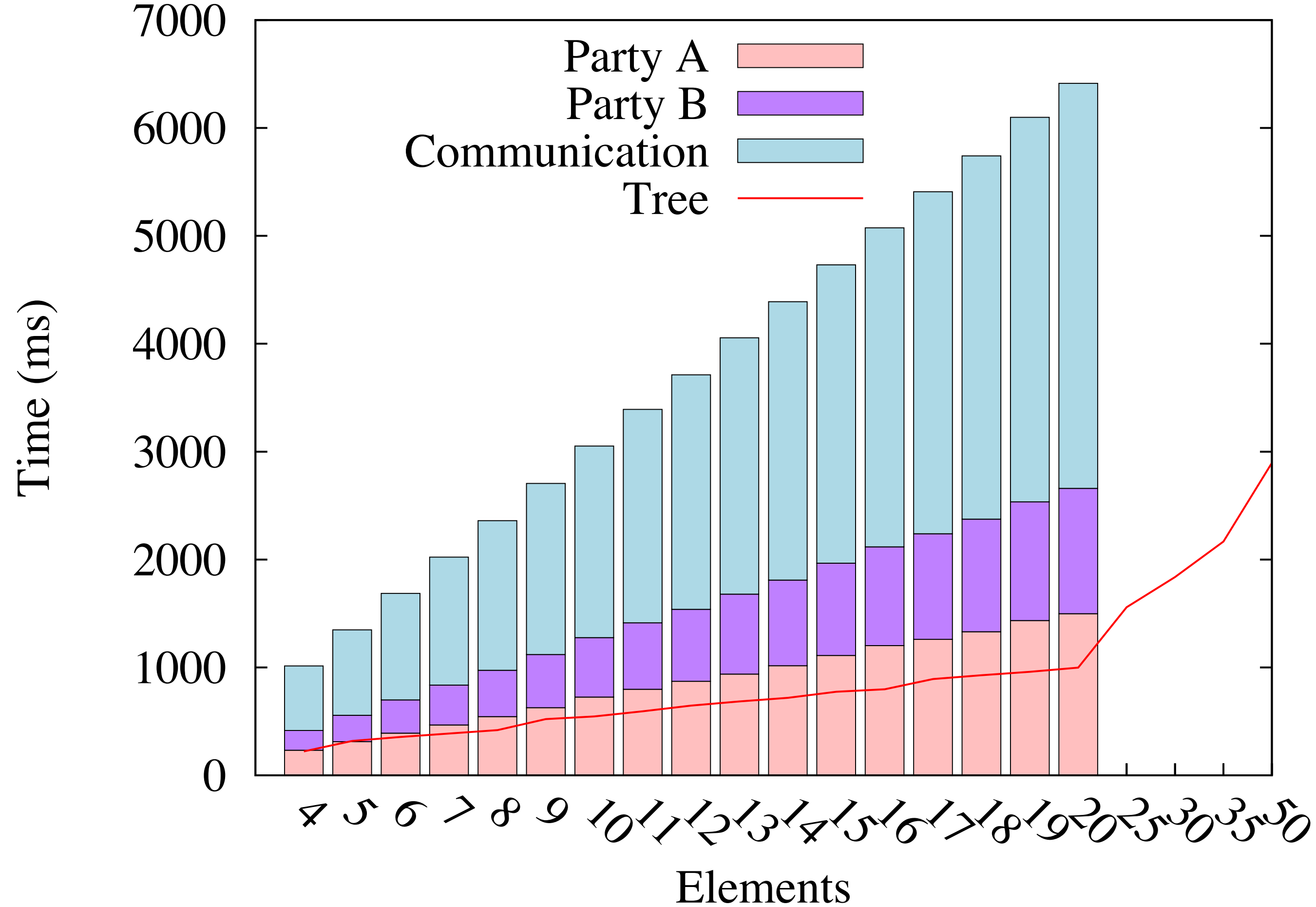
  or in parallel

Argmax

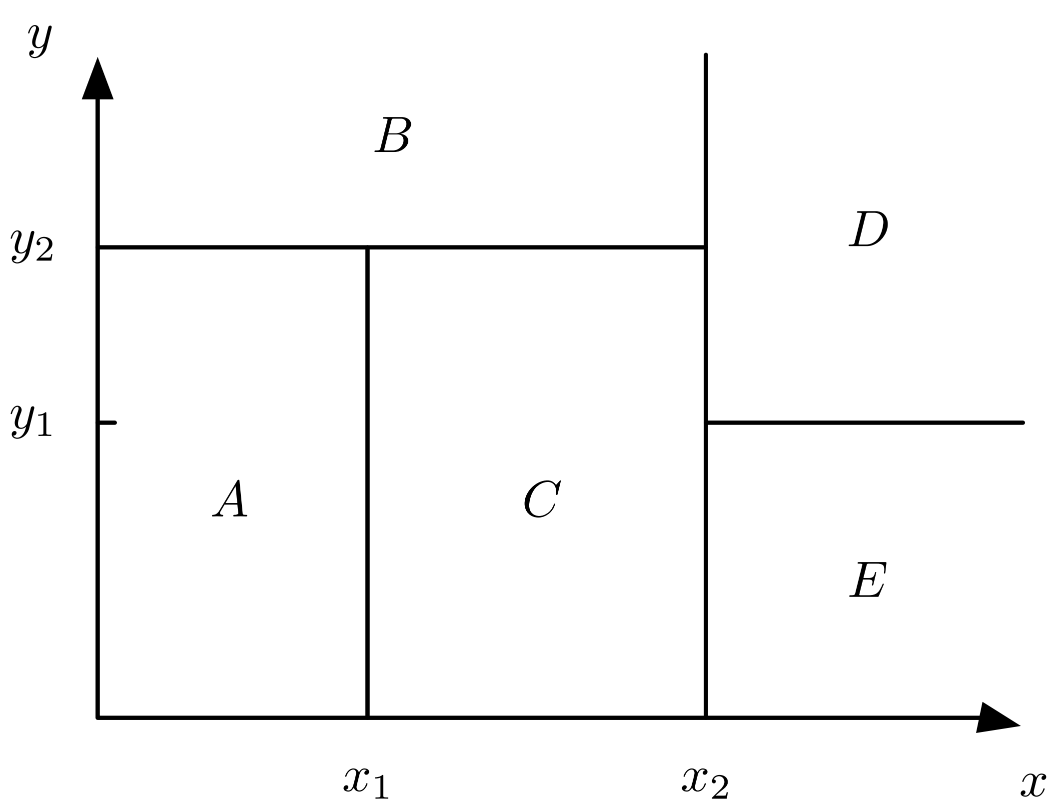
- Protocol : n-1 Compare & Swap

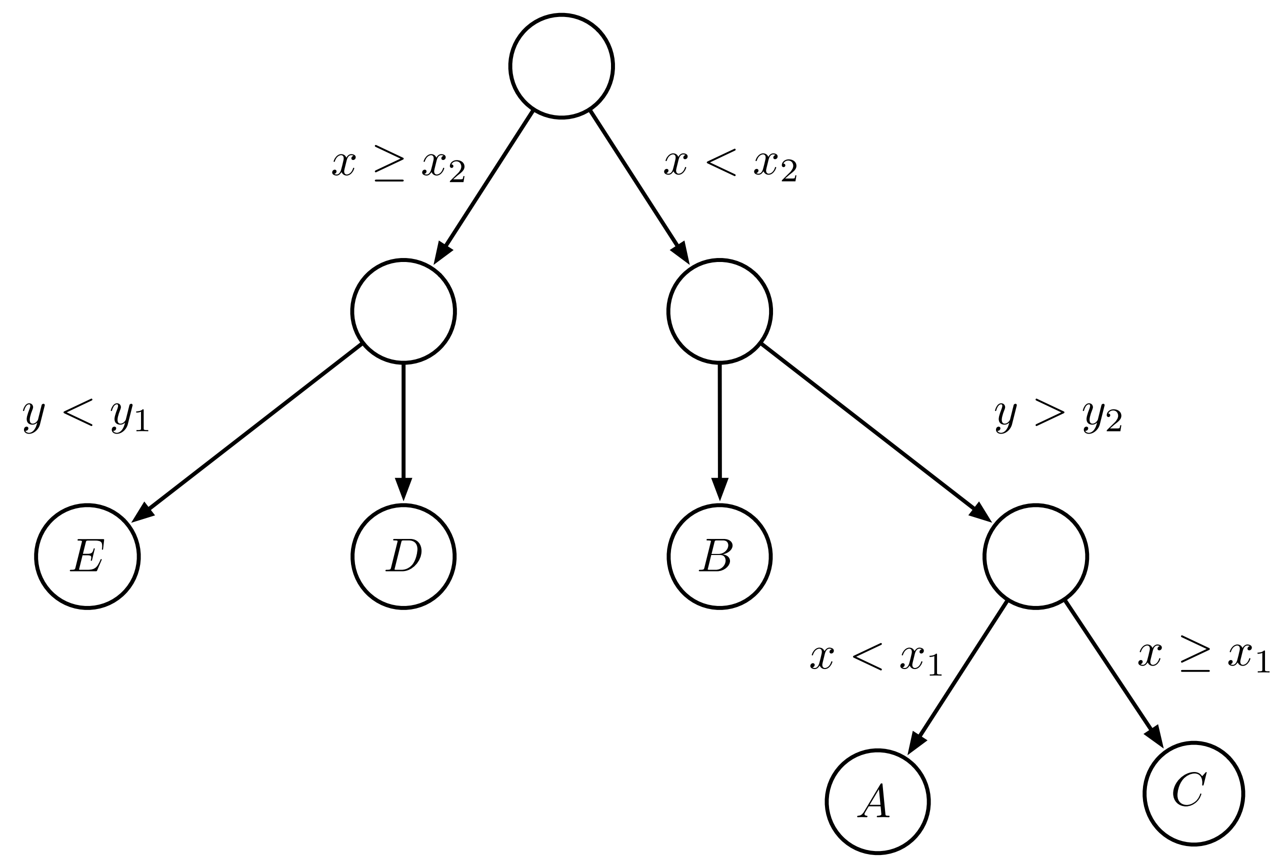
  sequentially

  or in parallel

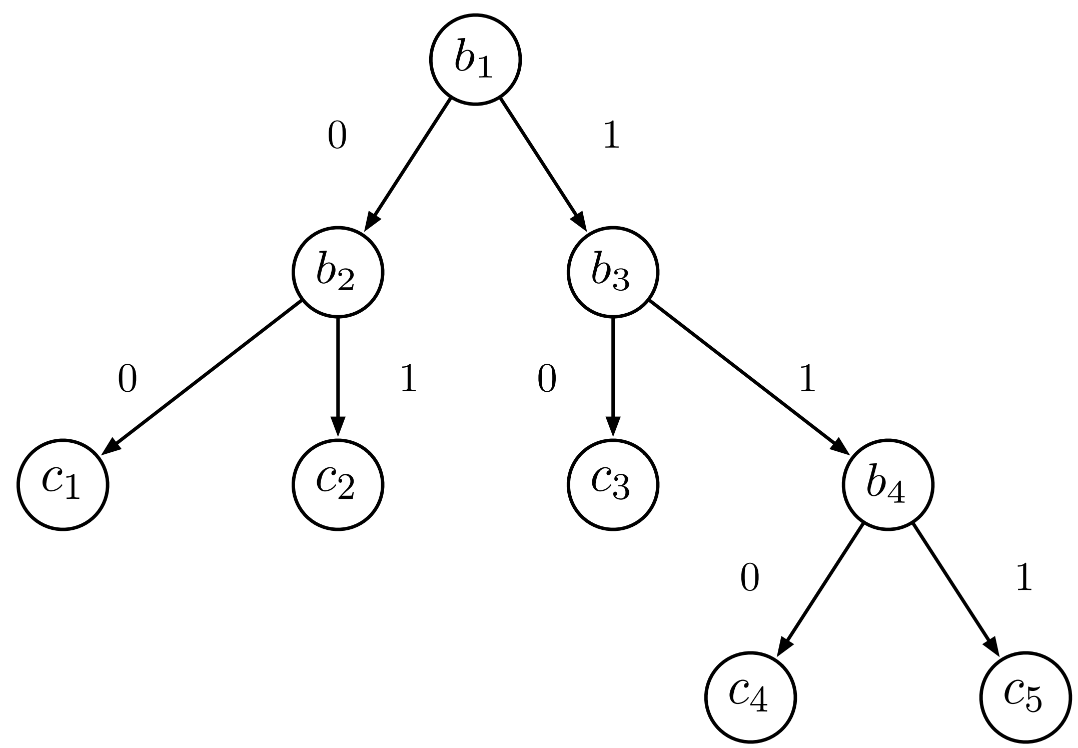


Decision Trees





Decision Trees



$$
\begin{aligned}
P(b_1, b_2, b_3, b_4, c_1, \ldots, c_5) = \quad & b_1 \cdot (b_3 \cdot (b_4 \cdot c_5 + (1 - b_4) \cdot c_4) + (1 - b_3) \cdot c_3) \\
& +(1 - b_1) \cdot (b_2 \cdot c_2 + (1 - b_2) \cdot c_1)
\end{aligned}
$$

Decision Trees

$$P(b_1, b_2, b_3, b_4, c_1, \ldots, c_5) = \quad b_1 \cdot (b_3 \cdot (b_4 \cdot c_5 + (1 - b_4) \cdot c_4) + (1 - b_3) \cdot c_3)$$
$$+ (1 - b_1) \cdot (b_2 \cdot c_2 + (1 - b_2) \cdot c_1)$$

- Polynomial evaluation

  Leveled Homomorphic Encryption

- Binary Variables
- Binary Coefficients ! (SIMD)

Efficient LHE

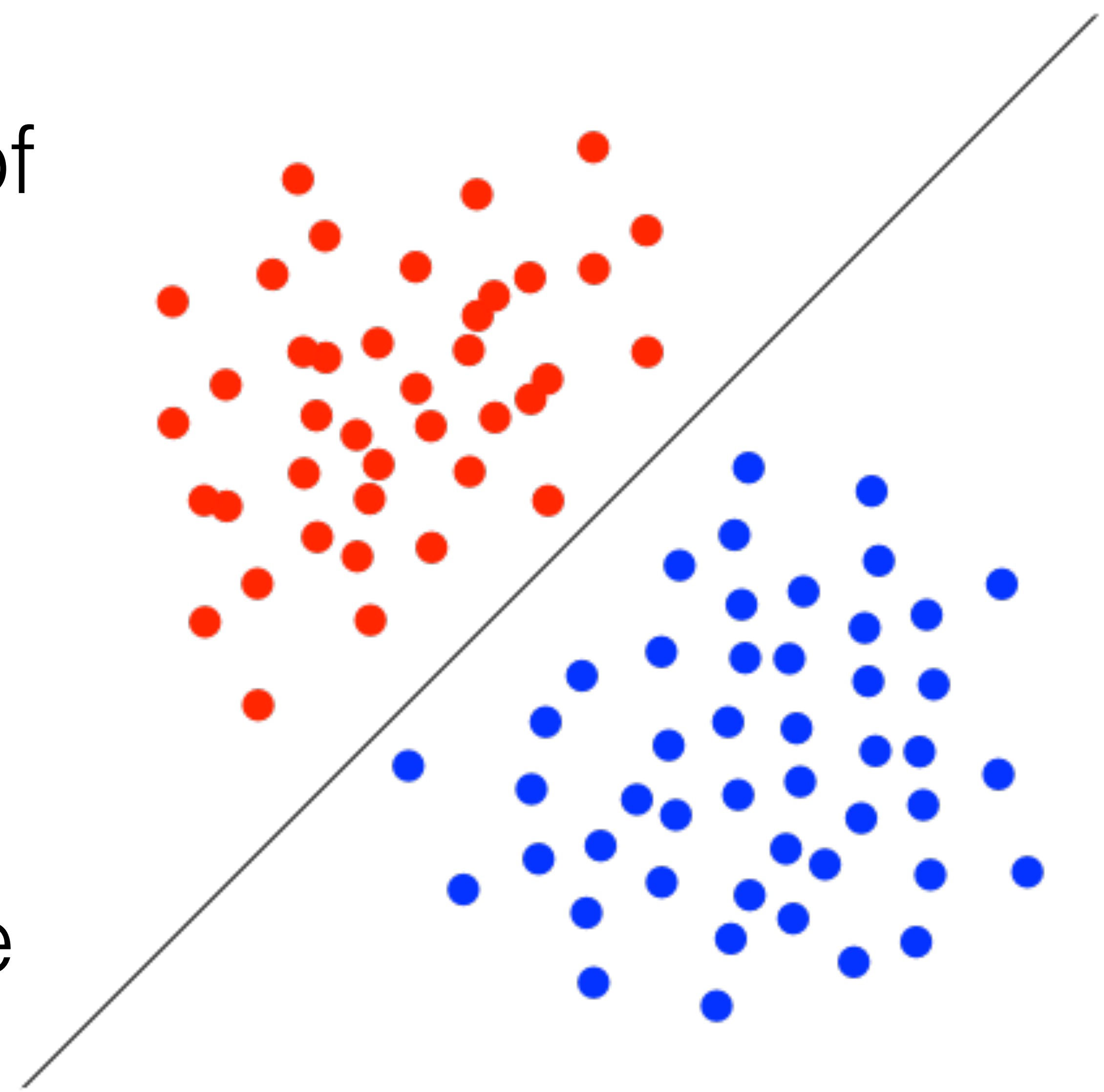
Classifiers

In Practice

- Linear Classifier
- Naïve Bayes Classifier
- Decision Trees

Linear Classifier

- Separate two sets of points
- Very common classifier
- Dot product + Encrypted compare



Linear Classifier

Model Size	Computation		Time / protocol		Total	Comm.	Inter.
	Client	Server	Dot Product	Enc. Comp.			
30	46.4 ms	43.8 ms	194 ms	9.67 ms	204 ms	35.84 kB	7
47	55.5 ms	43.8 ms	194 ms	23.6 ms	217 ms	40.19 kB	7

Evaluation on UC Irvine ML databases
40 ms network latency
2,66 GHz Intel Core i7

Naïve Bayes Classifier

Naïve Bayes Classifier

- Classification $\operatorname*{argmax}_{i \in [k]} p(C = c_i | X = x)$

Naïve Bayes Classifier

- Classification $\underset{i \in [k]}{\operatorname{argmax}} \, p(C = c_i | X = x)$
- Bayes Formula $\underset{i \in [k]}{\operatorname{argmax}} \, \frac{p(C = c_i, X = x)}{p(X = x)}$

Naïve Bayes Classifier

- Classification $\underset{i \in [k]}{\operatorname{argmax}}\, p(C = c_i | X = x)$
- Bayes Formula $\underset{i \in [k]}{\operatorname{argmax}}\, p(C = c_i, X = x)$

Naïve Bayes Classifier

- Classification $\operatorname*{argmax}_{i \in [k]} p(C = c_i | X = x)$
- Bayes Formula $\operatorname*{argmax}_{i \in [k]} p(C = c_i, X = x)$
- Naïve Model $\operatorname*{argmax}_{i \in [k]} p(C = c_i, X_1 = x_1, \ldots, X_d = x_d)$

Naïve Bayes Classifier

- Classification $\underset{i \in [k]}{\operatorname{argmax}}\, p(C = c_i | X = x)$
- Bayes Formula $\underset{i \in [k]}{\operatorname{argmax}}\, p(C = c_i, X = x)$
- Naïve Model $\underset{i \in [k]}{\operatorname{argmax}}\, p(C = c_i) \prod_{j=1}^{d} p(X_j = x_j | C = c_i)$

Naïve Bayes Classifier

- Classification $$\operatorname*{argmax}_{i \in [k]} p(C = c_i | X = x)$$
- Bayes Formula $$\operatorname*{argmax}_{i \in [k]} p(C = c_i, X = x)$$
- Naïve Model $$\operatorname*{argmax}_{i \in [k]} p(C = c_i) \prod_{j=1}^{d} p(X_j = x_j | C = c_i)$$

Naïve Bayes Classifier

- Classification $\underset{i \in [k]}{\operatorname{argmax}}\, p(C = c_i | X = x)$
- Bayes Formula $\underset{i \in [k]}{\operatorname{argmax}}\, p(C = c_i, X = x)$
- Naïve Model $\underset{i \in [k]}{\operatorname{argmax}}\, p(C = c_i) \prod_{j=1}^{d} p(X_j = x_j | C = c_i)$

$$\underset{i \in [k]}{\operatorname{argmax}} \quad \log p(C = c_i) \sum_{j=1}^{d} \log p(X_j = x_j | C = c_i)$$

Naïve Bayes Classifier

k	d	Computation Client	Computation Server	Running Time	Comm.	Inter.
2	9	150 ms	104 ms	479 ms	72.47 kB	14
5	9	537 ms	368 ms	1415 ms	150.7 kB	42
24	70	1652 ms	1664 ms	3810 ms	1911 kB	166

Evaluation on UC Irvine ML databases
40 ms network latency
2,66 GHz Intel Core i7

Decision Tree

- Combination of other classifiers
- In this example, linear classifiers
- Linear classifier + ES Switching + Decision Trees

Decision Tree

Tree Specs.		Computation		Time / Protoc.		FHE		Com m.	Inter.
N	D	Client	Server	Lin. Class.	ES Switch	Eval	Decrypt		
4	4	1579 ms	798 ms	446 ms	1639 ms	239 ms	33.51 ms	2639 kB	30
6	4	2297 ms	1723 ms	1410 ms	7406 ms	899 ms	35.1 ms	3555 kB	44

Evaluation on UC Irvine ML databases
40 ms network latency
2,66 GHz Intel Core i7

In conclusion

- Composable building blocks for secure classifiers
- Practical performances

Future work :

- Less roundtrips (work on the protocols)
- More parallelism (work on the implementation)

Questions?