CRYPTOGRAPHY

(lecture 10)

Literature:

“Lecture Notes on Cryptographic Protocols” (ch **5.0,5.1,5.2.2,5.2.4**, all ch 5)
“Efficient Secure Two-Party Protocols” by C. Hazay & H. Lindell (ch2, 6)

Announcements

- Exercise session Dec 6 (8-9:45): by **William** (last)
- Lecture on Dec 6th (10-11:45): by **Elena** on advanced and fun things + Q&A
- Lecture on Dec 9th (10-11:45): by **Victor** on ABC [also zoom streaming]
- Lecture on Dec 13th (10-11:45): by **Elena** course recap + exercises + exam template
- **No** exercises/lectures (8-9:45) on Dec 9th, Dec 13th
- Office Hours by **Ivan** on weeks 48-49-50 on Wednesdays 13:00-14:00 and Fridays: 16:00-17:00 in 3128 and Zoom on demand
- **Video** `Signal_Protocol.mp4` on part of Lecture 8 available on Canvas (Module 2) ⚠
- Lecture 9, updated slide **19** and **new slide 20**

for a different course

Module 3: Agenda

Introduction to MPC

Commitment Schemes (Pedersen) **HA3**

(Verifiable) Secret Sharing (Shamir) **HA3**

Oblivious Transfer

MPC Security

- The Real/Ideal World Paradigm

Zero-Knowledge Proofs

- Intuition
- Ideal Functionality
- Interactive ZK Proofs

Σ (Sigma) Protocols

- Syntax
- Schnorr (Knowledge of dLog) - Proof
- Chaum-Pedersen (Same dLog) **HA3**
- Compound Statements (OR, AND) - Proof
- Knowledge of Pedersen Commitments

Removing Interaction

- Fiat-Shamir Heuristic **HA3**

Generic 2 Party Computation

- Garbled Circuits
- Yao’s Two Party Protocol

Syntax for Multi-Party Computation Protocols (MPC)

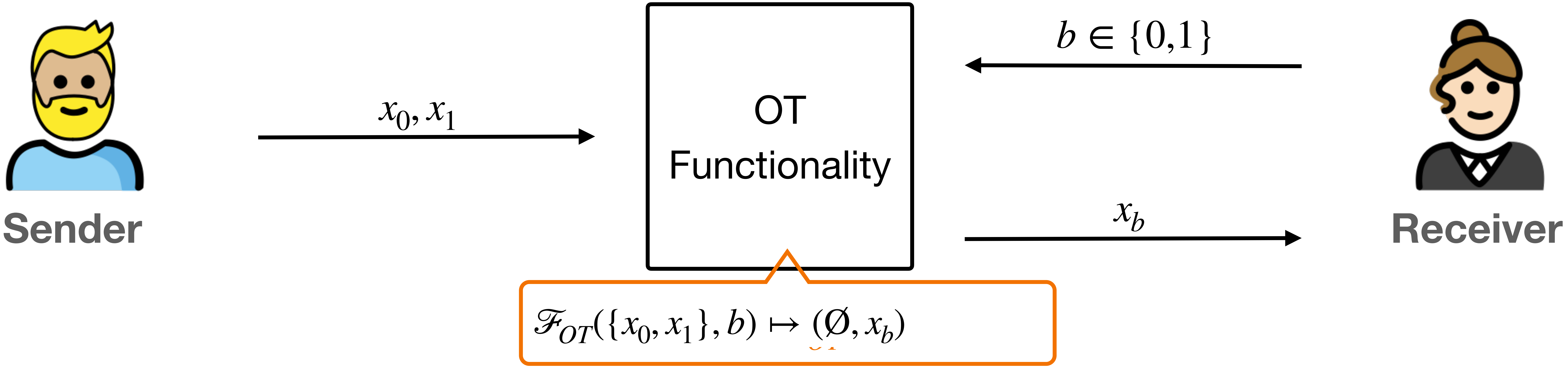
Syntax Any multiparty computation protocol Π among n parties is defined by specifying a process that maps n-tuples of inputs (one for each party) to n-tuples of outputs (one for each party). Formally this process is called **ideal functionality** and is denoted by

$$\mathscr{F} : \{0,1\}^* \times \{0,1\}^* \cdots \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^* \cdots \times \{0,1\}^*$$

$$\mathscr{F}(x_1, x_2, \ldots, x_n) \quad = \quad (\ f_1(x_1, \ldots, x_n)\ , \ldots,\ f_n(x_1, \ldots, x_n)\ )$$

$$(\quad y_1 \quad , \quad y_2 \quad , \ldots\ldots, \quad y_n \quad)$$

Formalizing Security Notions for MPC



Sender

x_0, x_1

OT
Functionality

$b \in \{0,1\}$

x_b

Receiver

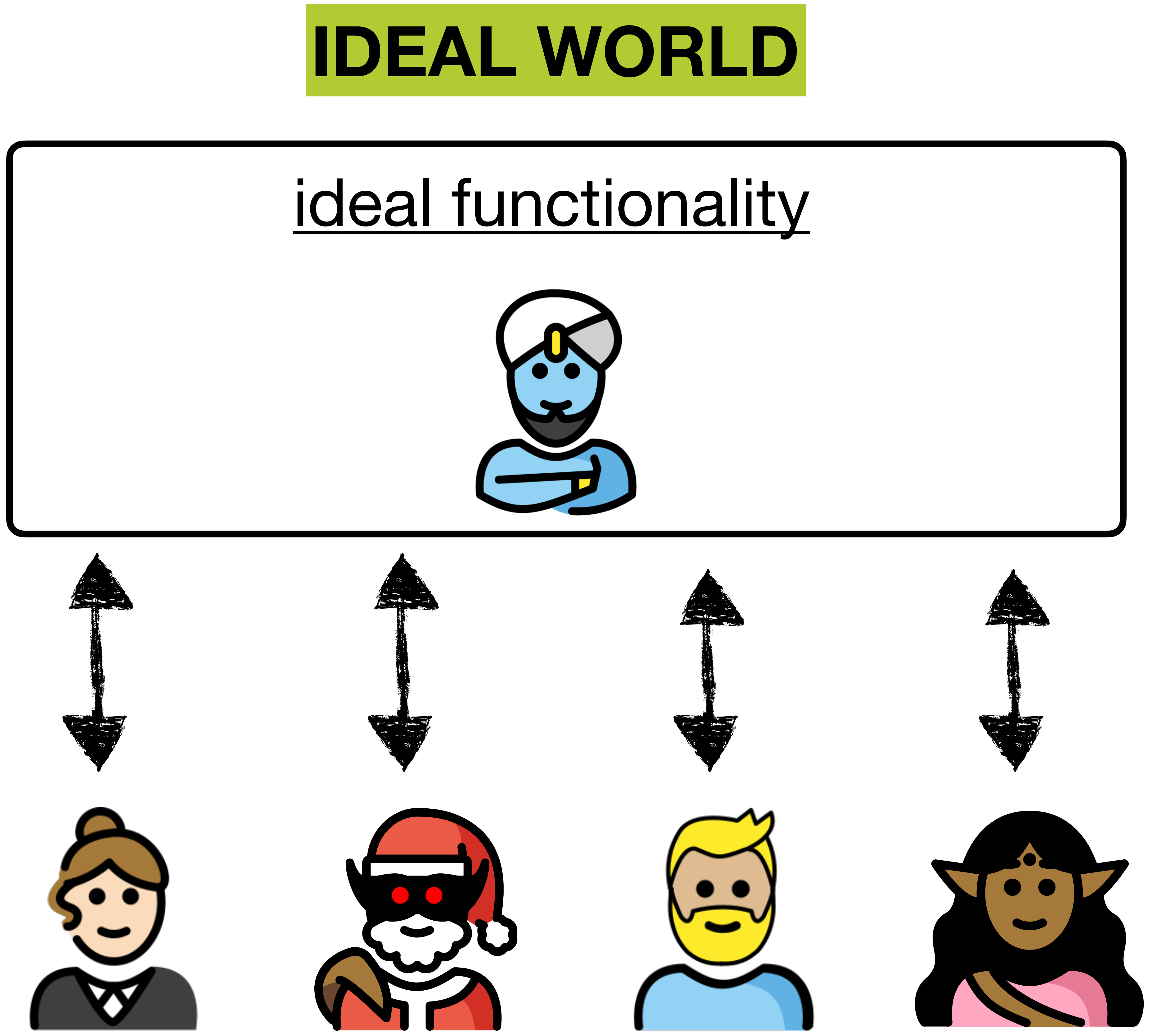
$\mathcal{F}_{OT}(\{x_0, x_1\}, b) \mapsto (\emptyset, x_b)$

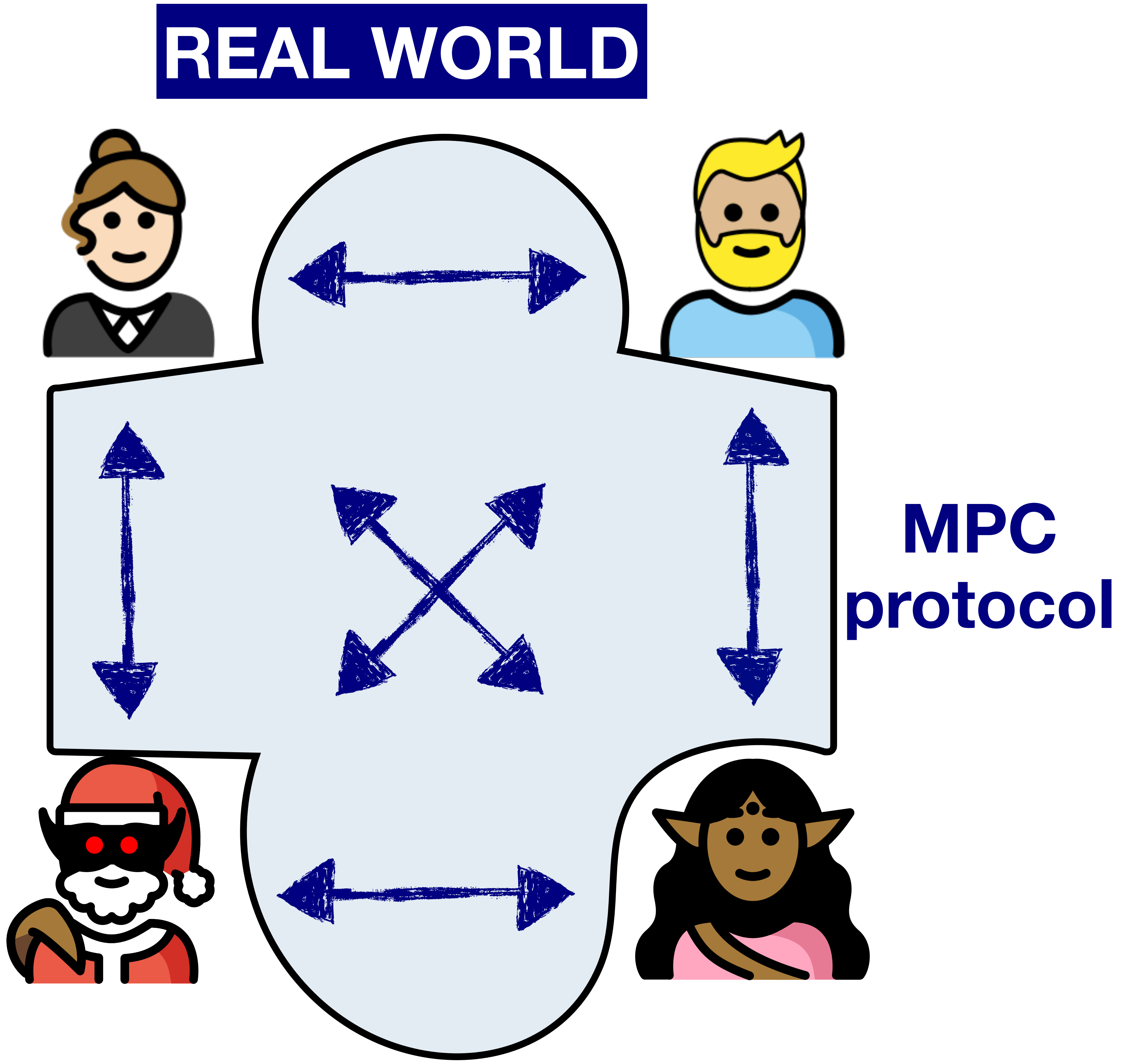
Privacy: No party should learn anything more than its prescribed output.

How can we model “a party learns nothing”?

whatever can be computed by a party participating in the protocol can be computed based on its input and output only

Ideal World Vs Real World (Security Paradigm)



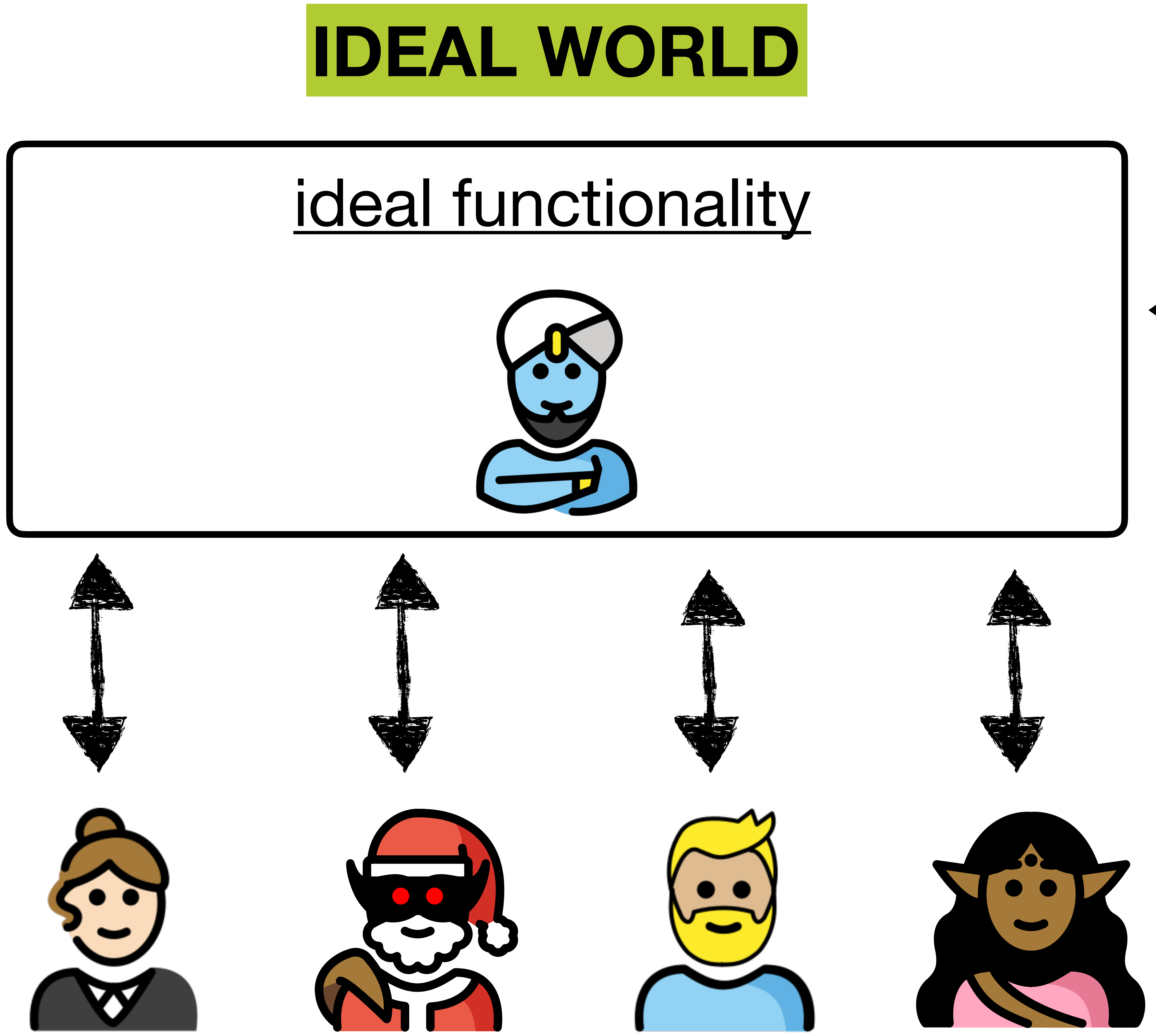


An MPC protocol allows multiple parties to **jointly evaluate a specific function** over the parties' **private inputs**

The goal of an MPC protocol is to provide **security in the real world** (given a set of *assumptions*) **that is equivalent to that in the ideal world.**

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DzGLx_mLLu0c&psig=AOvVaw2apYsTgLReXypLHAZ3jP-3&ust=1650024734701000&source=images&cd=vfe&ved=2ahUKEwixrN2iw5P3AhXYZfEDHcXCDJcQjhx6BAgAEAw

The Ideal World



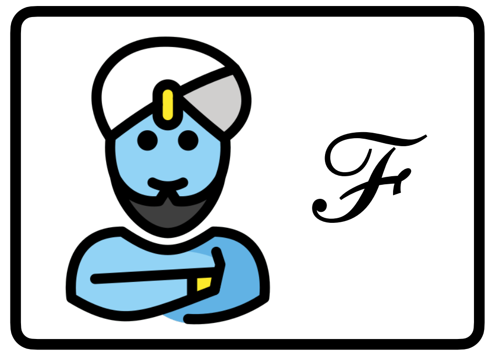
IDEAL WORLD

ideal functionality

A **fully trusted third party** carries out the computation of $\mathcal{F}(x_1, \ldots, x_n)$ and distributes the to each party its designated outcome

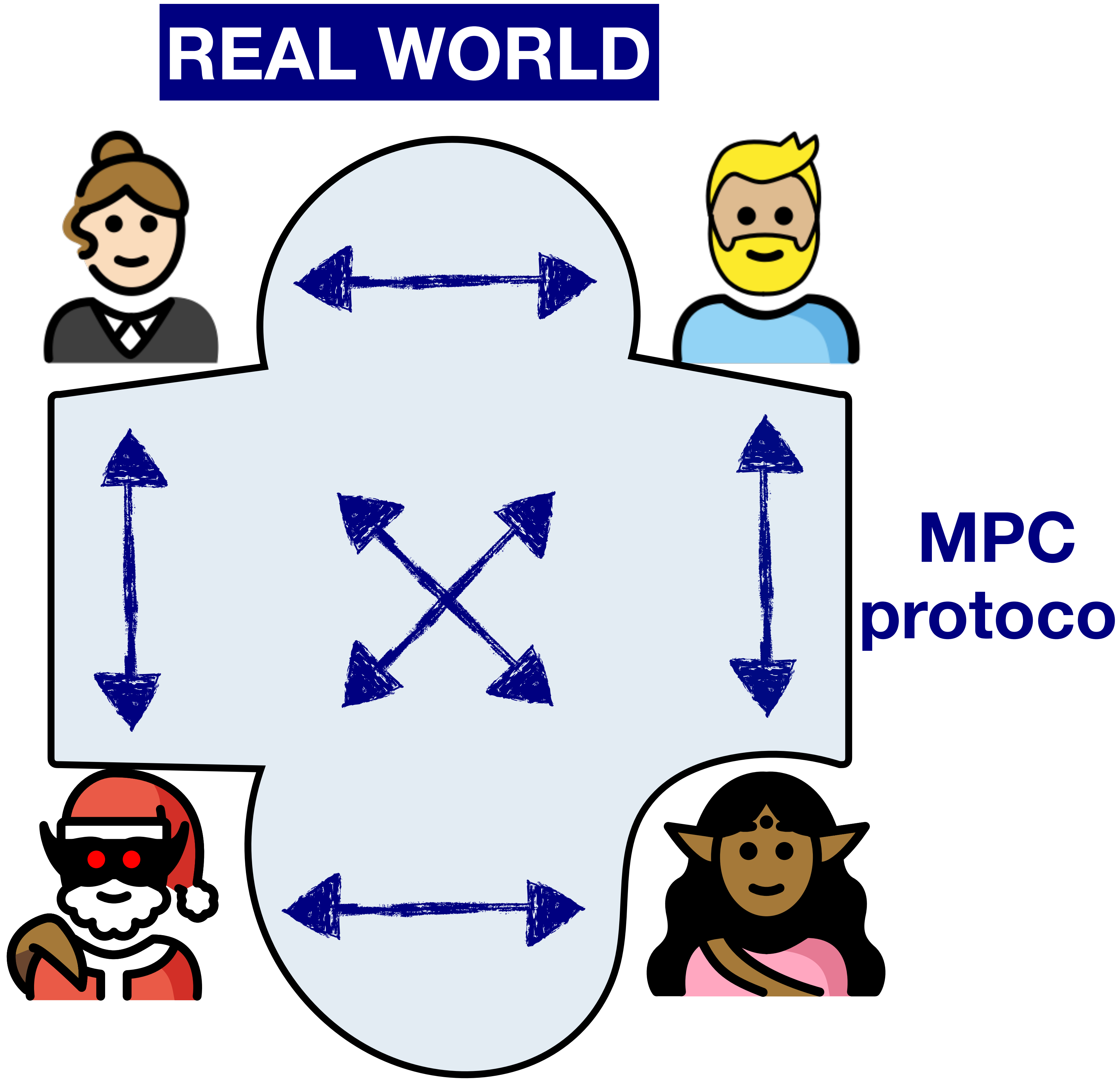
All communication channels are perfect (authenticated, confidential, noise-free)

$\mathcal{A}$ can take control over any subset of the parties

In reality, for cryptographers there is no trusted party  but we can use this *ideal* world as a *benchmark* against which to judge the security of an actual protocol.

The Real World

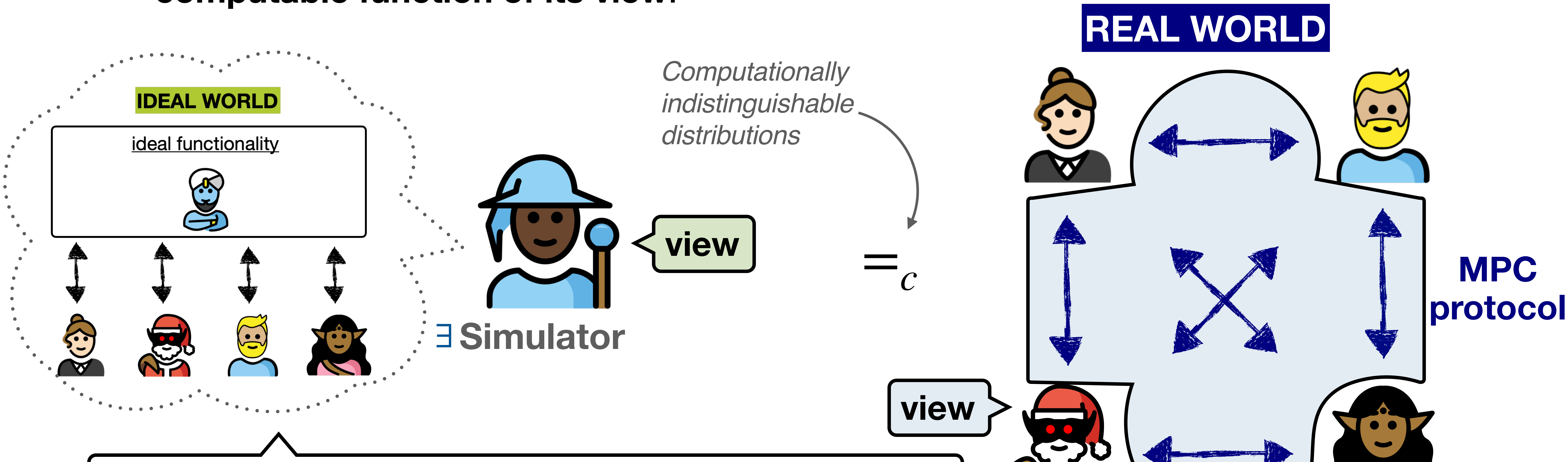
The **trusted party** is replaced by an MPC protocol $\Pi_{\mathscr{F}}$. For each party, $\Pi_{\mathscr{F}}$ specifies a “next-message” function that takes as input sec.par, x_i, some randomness r_i, and the list of messages received so far; the “next-message” function outputs either a message and addressee, or else instructions to recover the party’s output.



The real world protocol $\Pi_{\mathscr{F}}$ is considered secure if **any effect** that **any adversary** can achieve in the **real world** can also be achieved by **a corresponding adversary in the ideal world**.

Defining Security in the Real-Ideal World Paradigm

- The **view of a party** consists of its private input, its random tape, and the list of all messages received during the protocol.
- The **view of an adversary** consists of the combined views of all corrupt parties.
- Anything an adversary **learns** from running the protocol must be an **efficiently computable function of its view**.



Sim's view in the ideal world consists of all inputs sent to the ideal functionality and outputs receives from it.

MPC Security Against Semi-Honest Adversaries

A protocol is secure against semi-honest adversaries if the corrupted parties in the real world have views that are indistinguishable from their views in the ideal world.

MPC Security A protocol Π securely realizes the functionality $\mathscr{F}$ in the presence of **semi-honest adversaries** if there exists a simulator Sim such that, **for every** subset of corrupted parties $C \subseteq \{1,2,\ldots,n\}$ and **all** inputs $x_1, \ldots, x_n$, it holds that

$$Real_{\Pi}(\lambda, C; x_1, \ldots, x_n) =^c Ideal_{\mathscr{F},\mathsf{Sim}}(\lambda, C; x_1, \ldots, x_n)$$

i.e., the real and ideal views are computationally indistinguishable in the security parameter $\lambda \in \mathbb{N}$.

🧐 No $\mathscr{A}$ in here?

Security Against Malicious Adversaries

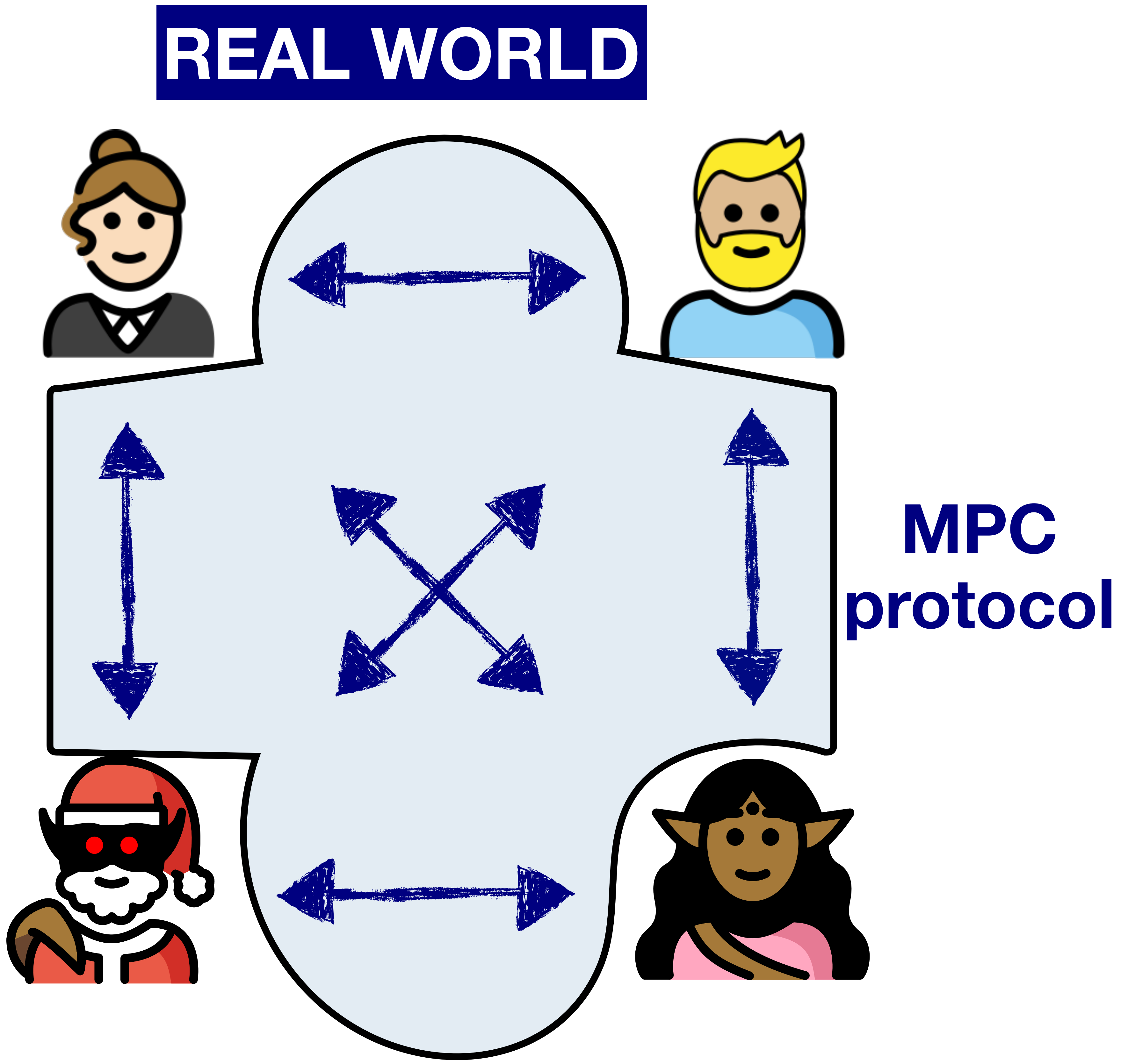
aka ***Active***: is a semi-honest adversary who **additionally** *may deviate arbitrarily* from the prescribed protocol in an attempt to violate security

how does this affect the output of honest parties?

$\mathscr{A}$ may send *inconsistent* messages to different parties

what can we guarantee about the output of corrupted parties?

how can we set the input of corrupted parties in Ideal ?



Module 3: Agenda

Introduction to MPC
Commitment Schemes (Pedersen)
(Verifiable) Secret Sharing (Shamir)
Oblivious Transfer

MPC Security

- The Real/Ideal World Paradigm

Nollkunskapsbevis

Zero-Knowledge Proofs

- Intuition
- Ideal Functionality
- Interactive ZK Proofs

Σ (Sigma) Protocols

- Syntax
- Schnorr (Knowledge of dLog) - Proof
- Chaum-Pedersen (Same dLog)
- Compound Statements (OR, AND) - Proof
- Knowledge of Pedersen Commitments

Removing Interaction

- Fiat-Shamir Heuristic

Generic 2 Party Computation

- Garbled Circuits
- Yao’s Two Party Protocol

Interactive Proofs

You can't have your cake and eat it too!

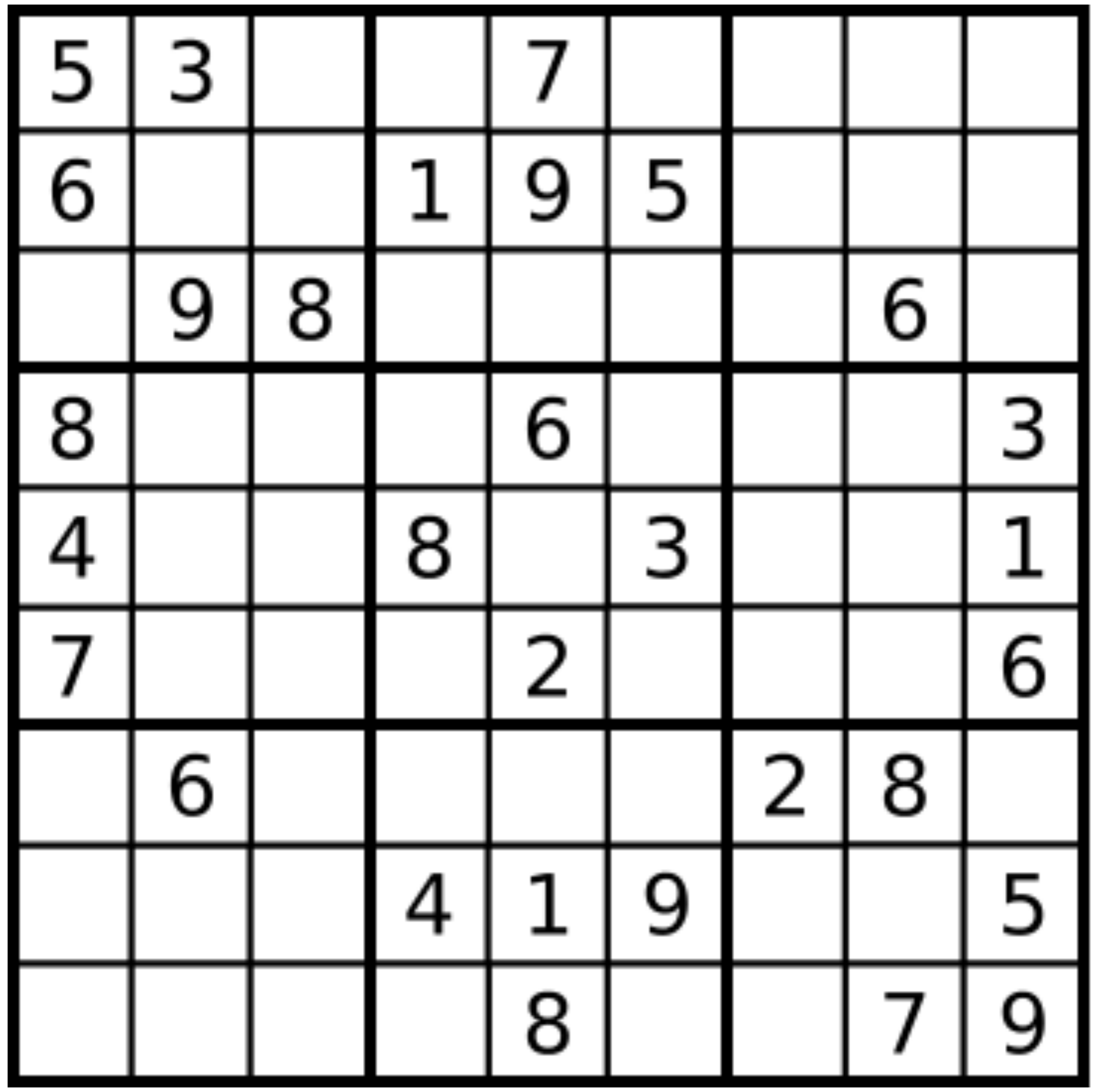
well, with crypto you do :)

zero knowledge proofs

Zero Knowledge Proofs - a Metaphor

Intuitively: a protocol is zero-knowledge if it communicates exactly the knowledge that was intended, and *no extra* (zero) knowledge.

How To Formalise This Into Math/Crypto?



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

`x`

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

`w`

set of all valid sudoku starters

solution satisfies sudoku rules

The most general formalisation: `x ∈ L iff ∃ w s.t. R(x,w) = 1`

How To Formalise This Into Math/Crypto?

Definition: Zero Knowledge Proof System

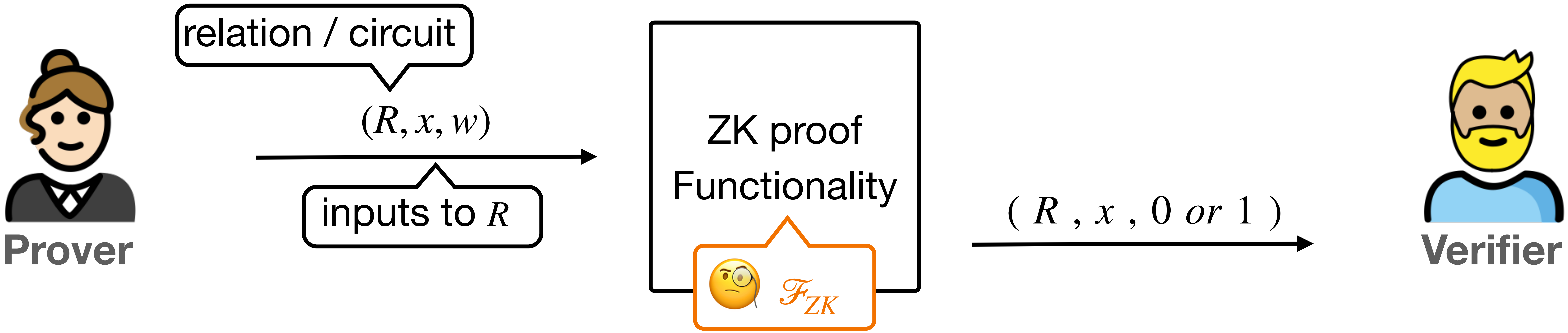
A zero-knowledge (ZK) proof system is a process in which a Prover probabilistically convinces a verifier of the correctness of a mathematical proposition, and the verifier learns nothing else.

	statement	witness
factoring:	$\exists\ p,q$ s.t. $N = pq \wedge p,q$ primes	(p,q)
dLog:	$\exists\ sk$ s.t. $pk = g^{sk}$	sk
circuit satisfiability:	$\exists\ w$ s.t. $C(w)=1$	w

the language `L` is usually *implicit* in the application, what we will make explicit is the **relation** `R`

Zero Knowledge Proof (Ideal Functionality)

The prover claims to know a *witness* w such that the *relation* $R(x, w) = 1$ holds for the *statement* x



relation / circuit

(R, x, w)

inputs to R

ZK proof Functionality

$\mathcal{F}_{ZK}$

$(R , x , 0 \; or \; 1)$

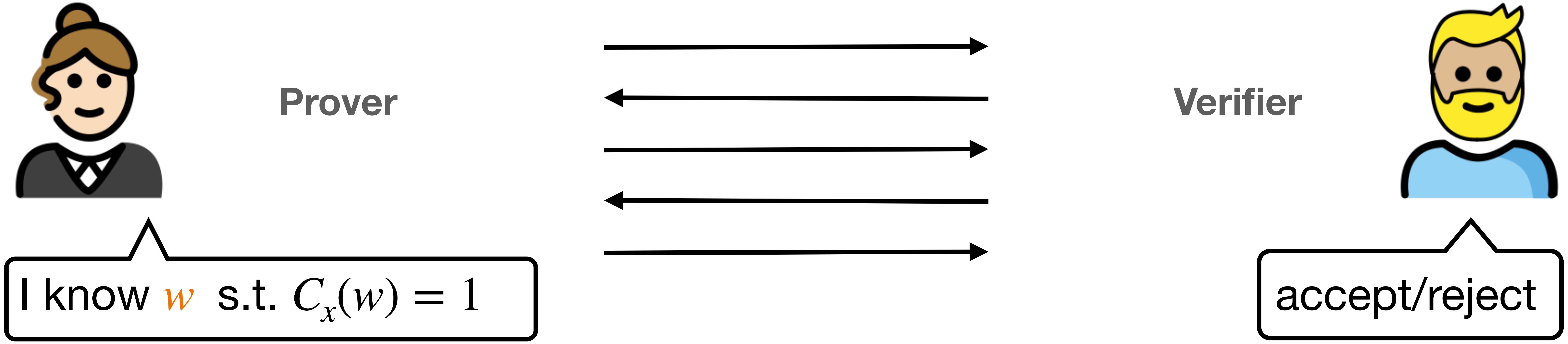
Prover

Verifier

$$\mathcal{F}_{ZK}(\{R, x, w\}, \emptyset) \mapsto (\emptyset, \{R, x, \text{Bool}[R(x, w) = 1]\})$$

Prover learns nothing, Verifier only learns whether w satisfies $R(x, \cdot\,)$, but nothing else about the secret w

Interactive ZK Proof - Syntax



Prover

Verifier

I know w s.t. $C_x(w) = 1$

accept/reject

Syntax: A Zero Knowledge proof (Π_{ZK}) for a relation R represented as a circuit C_x, is an **interactive protocol** between a Prover (P) and a Verifier (V) that realizes the following:

- **Input**: P and V know a circuit $C_x : \{0,1\}^n \rightarrow \{0,1\}$.
  In addition, P knows a secret input $w \in \{0,1\}^n$ to C_x
- **Output**: V learns whether $C_x(w) = 1$ (i.e., whether P knows an input w that satisfies the circuit $C_x(\cdot) = R(x, \cdot)$)

ZK Proof - Properties

This property is sometimes called ‘completeness'

Correctness: If P knows w s.t. $C_x(w) = 1$, then at the end of the Π_{ZK} protocol, V will reject only with negligible probability

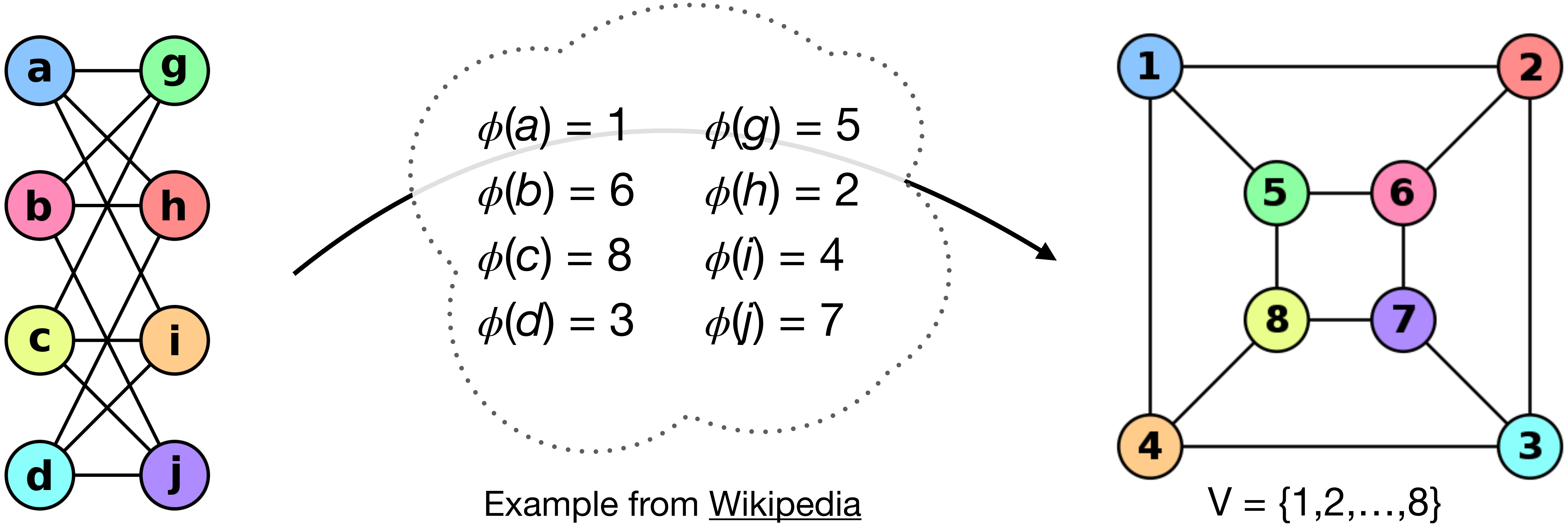
Soundness: If a cheating prover P* does not know a valid witness w, then at the end of the Π_{ZK} protocol, V will accept only with negligible probability

Zero-Knowledge: Once the protocol is completed, V learns nothing about w

$\mathrm{P}* = \mathscr{A}$ *is malicious*

A Prime Example: ZK Proof for Graph Isomorphism

A (undirected) **graph** is a pair $\mathcal{G}$ = (V,E), where V is a set of nodes (called vertexes) and E is a binary symmetric relation on V (identifying the edges of the graph).



$\phi(a) = 1$

$\phi(b) = 6$

$\phi(c) = 8$

$\phi(d) = 3$

$\phi(g) = 5$

$\phi(h) = 2$

$\phi(i) = 4$

$\phi(j) = 7$

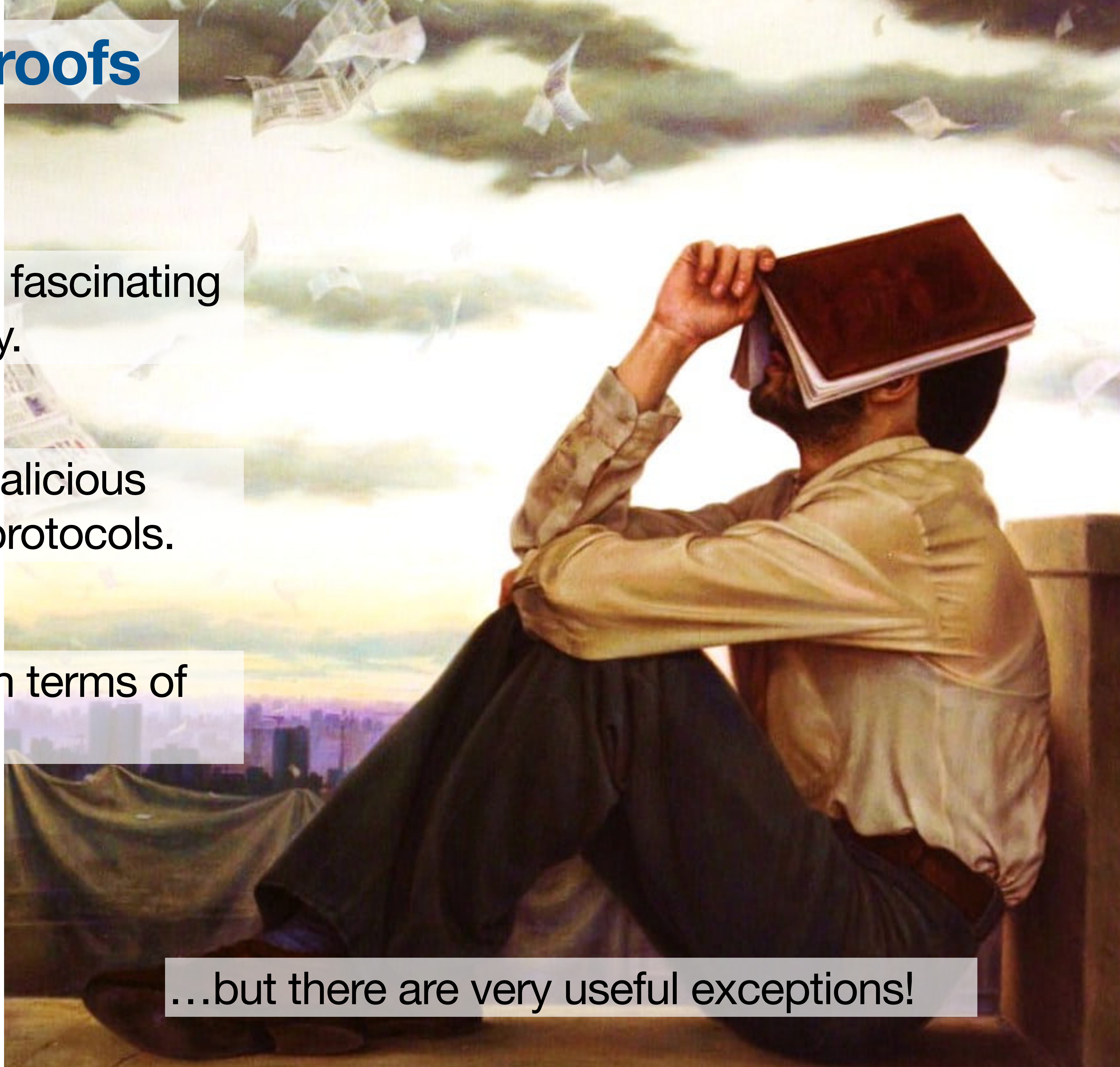
Example from Wikipedia

V = {1,2,…,8}

E = {(1,2), (1,5), (1,4), (2,6)…}

A graph **isomorphism** ϕ between (V,E) and (V’,E’) is a bijection $\phi : \mathcal{G} \to \mathcal{G}'$ such that $(v_1, v_2) \in E \quad \mathit{iff} \quad (\phi(v_1), \phi(v_2)) \in E'$.

A Few Words About ZK Proofs



The theory of ZK Proofs is extremely fascinating and it is fundamental in cryptography.

ZK Proofs can be used to achieve malicious security in multi-party computation protocols.

In general ZK Proofs are expensive in terms of computation & communication.

…but there are very useful exceptions!

Module 3: Agenda

Introduction to MPC
Commitment Schemes (Pedersen)
(Verifiable) Secret Sharing (Shamir)
Oblivious Transfer

MPC Security

- The Real/Ideal World Paradigm

Zero-Knowledge Proofs

- Intuition
- Ideal Functionality
- Interactive ZK Proofs

Σ (Sigma) Protocols

A special case of interactive ZK proofs

- Syntax
- Schnorr (Knowledge of dLog) - Proof
- Chaum-Pedersen (Same dLog)
- Compound Statements (OR, AND) - Proof
- Knowledge of Pedersen Commitments

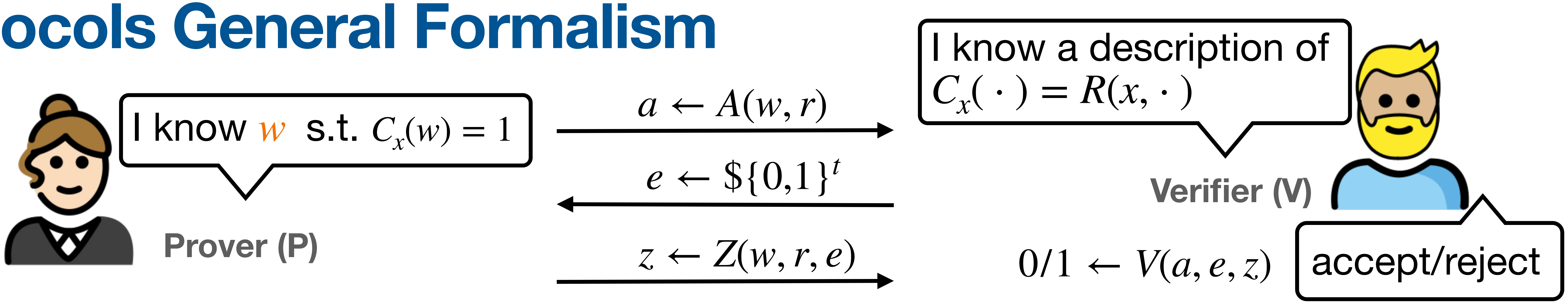
Removing Interaction

- Fiat-Shamir Heuristic

Generic 2 Party Computation

- Garbled Circuits
- Yao's Two Party Protocol

Σ Protocols General Formalism



I know w s.t. $C_x(w) = 1$

Prover (P)

$$\xrightarrow{a \leftarrow A(w, r)}$$

$$\xleftarrow{e \leftarrow \$\{0,1\}^t}$$

$$\xrightarrow{z \leftarrow Z(w, r, e)}$$

I know a description of $C_x(\cdot) = R(x, \cdot)$

Verifier (V)

$0/1 \leftarrow V(a, e, z)$

accept/reject

Definition A Σ-protocol for relation R if it is a three-move, public-coin protocol of the form depicted above that additionally satisfies the following requirements:

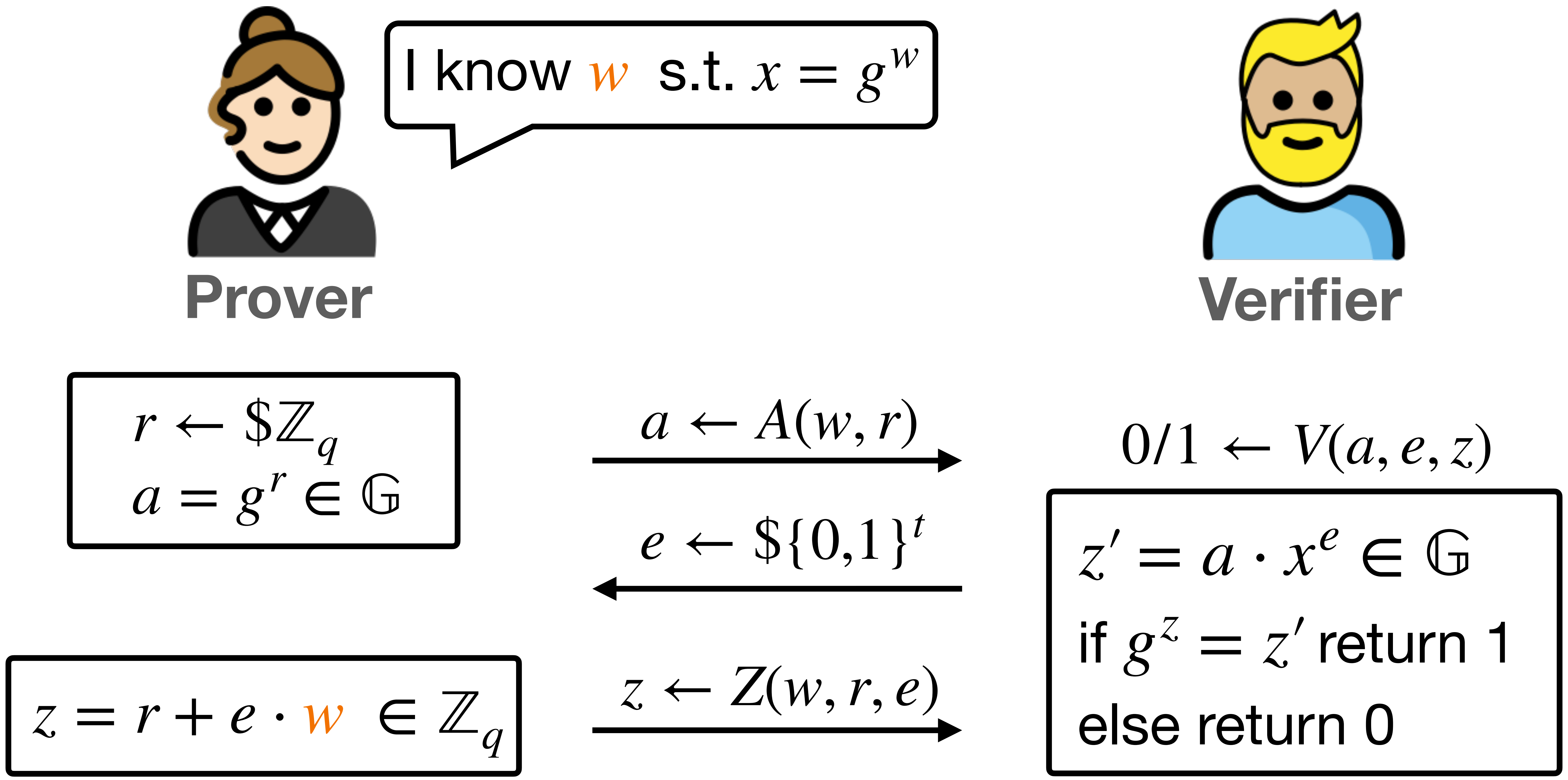
- **Completeness**: If P and V follow the protocol on input x and private input w to P where $(x, w) \in R$, then V always accepts.
- **Special soundness**: There exists a PPT algorithm $\mathscr{E}$ (extractor) that given any x and any pair of accepting transcripts $(a, e, z), (a, e', z')$ for x, with $e \neq e'$, outputs w such that $(x, w) \in R$.
- **Special honest verifier zero knowledge**: for every x and w such that $(x, w) \in R$ and every $e \in \{0,1\}^t$ it holds that

$$\{Sim(x, e)\} =_c \{\langle P(x, w), V(x, e)\rangle\}$$

where V 's random tape equals e and t is the challenge length.

Schnorr Σ-Protocol for Knowledge of dLog

Sometimes called Schnorr identification protocol



I know w s.t. $x = g^w$

Prover

Verifier

$r \leftarrow \$\mathbb{Z}_q$
$a = g^r \in \mathbb{G}$

$\xrightarrow{a \leftarrow A(w,r)}$

$\xleftarrow{e \leftarrow \$\{0,1\}^t}$

$z = r + e \cdot w \in \mathbb{Z}_q$

$\xrightarrow{z \leftarrow Z(w,r,e)}$

$0/1 \leftarrow V(a,e,z)$

$z' = a \cdot x^e \in \mathbb{G}$
if $g^z = z'$ return 1
else return 0

<u>public inputs</u> (available to both P and V)

- the description of a group $\mathbb{G}$ of prime order q with generator g
- the value $x \in \mathbb{G}$
- $R(x,w) : \mathbb{G} \times \mathbb{Z}_q \rightarrow \{0,1\}$, defined as $R(x,w) = 1 \ iff \ x = g^w$
- the challenge length $t = log_2(q) \in \mathbb{N}$

Completeness

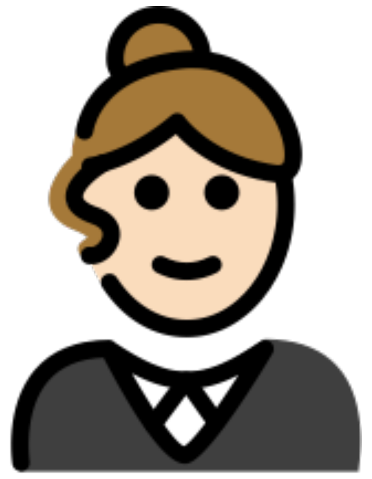
Special Soundness

Special HV ZK

whiteboard
proofs

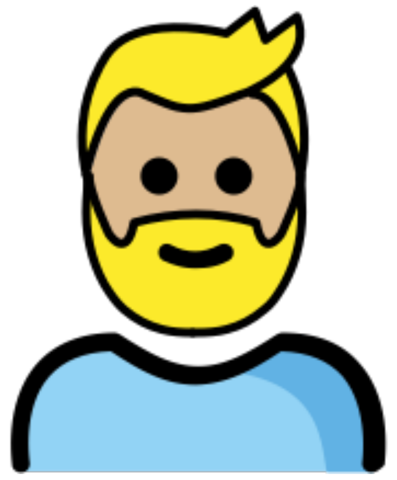
Chaum–Pedersen Σ-Protocol (Proof of Same dLog)

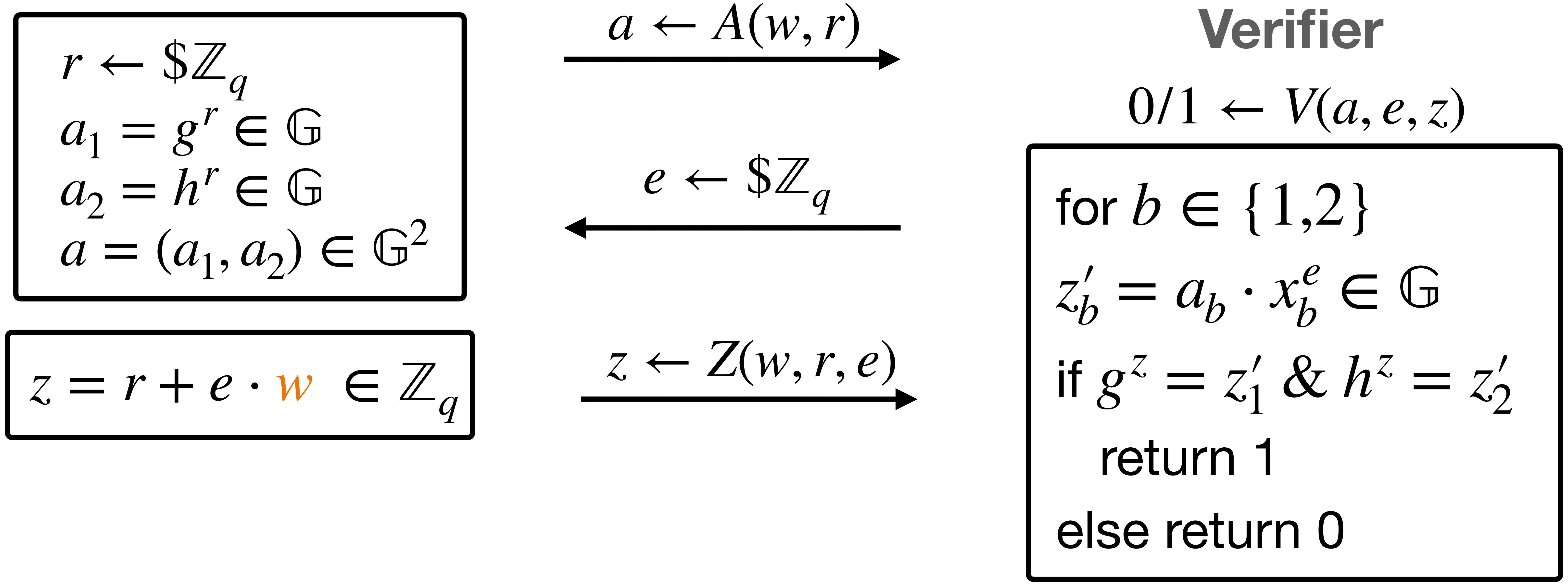
$$R = \{\big((\mathbb{G}, q, g, h, x_1, x_2)\ ,\ w\big) \mid g, h \in \mathbb{G}\ \&\ x_1 = g^w\ \&\ x_2 = h^w\}$$



$ZKPoK\{w : x_1 = g^w$ and $x_2 = h^w\}$

zero-knowledge proof of knowledge





Prover

$r \leftarrow \$\mathbb{Z}_q$
$a_1 = g^r \in \mathbb{G}$
$a_2 = h^r \in \mathbb{G}$
$a = (a_1, a_2) \in \mathbb{G}^2$

$\xrightarrow{a \leftarrow A(w, r)}$

$\xleftarrow{e \leftarrow \$\mathbb{Z}_q}$

$z = r + e \cdot w \ \in \mathbb{Z}_q$

$\xrightarrow{z \leftarrow Z(w, r, e)}$

Verifier

$0/1 \leftarrow V(a, e, z)$

```
for $b \in \{1,2\}$
$z'_b = a_b \cdot x_b^e \in \mathbb{G}$
if $g^z = z'_1$ & $h^z = z'_2$
    return 1
else return 0
```

🧐 Looks familiar? This solution is almost a parallel repetition of Schnorr
(except that the challenge(s) are now squashed into a single one for efficiency)

Chaum–Pedersen Σ-Protocol (Proof of Same dLog)

$$R = \{\left((\mathbb{G}, q, g, h, x_1, x_2)\ ,\ w\right) \mid g, h \in \mathbb{G}\ \&\ x_1 = g^w\ \&\ x_2 = h^w\}$$

🧐 what can we use this for?

$ZKPoK\{w : x_1 = g^w$ and $x_2 = h^w\}$

A

```
$r \leftarrow^{\$} \mathbb{Z}_q$
$a_1 = g^r \in \mathbb{G}$
$a_2 = h^r \in \mathbb{G}$
$a = (a_1, a_2) \in \mathbb{G}^2$
```

Z

```
$z = r + e \cdot w \ \in \mathbb{Z}_q$
```

V

```
for $b \in \{1,2\}$
$z'_b = a_b \cdot x_b^e \in \mathbb{G}$
if $g^z = z'_1$ & $h^z = z'_2$
    return 1
else return 0
```

Let $w = sk_A$ and $h = g^{sk_B}$ then this Σ-Protocol is a proof that (g, h, x_1, x_2) is a **Diffie-Hellman Tuple!**

Proving Compound Statements (AND, OR)

“AND” proofs

$$R_{dDH} = \{\big((\mathbb{G}, q, g, h, x_1, x_2)\ ,\ w\big) \,|\, g, h \in \mathbb{G}\ \&\ x_1 = g^w\ \&\ x_2 = h^w\}$$

make P prove both statements in parallel using a single challenge e for both proofs.

“OR” proofs

are a bit more complicated…

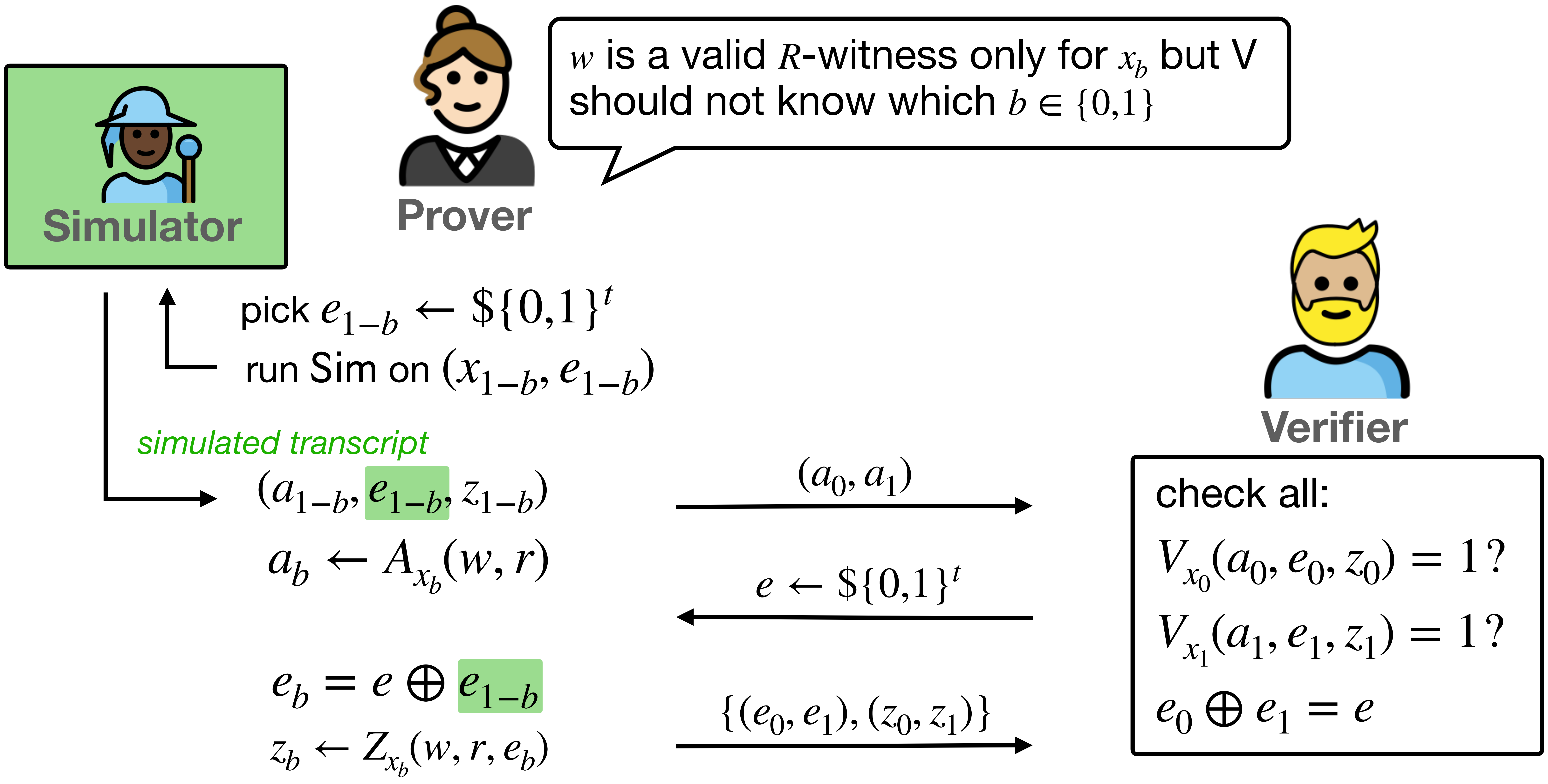
P wants to prove that: **either** $(x_0, w) \in R_0$ **OR** $(x_1, w) \in R_1$

ZK imposes to do this ***without revealing which is the case***

🧐 *The Trick* **:** P completes the protocol for the instance x_b that is true and “fakes” a proof for the other statement by running the simulator (in a clever way)

Proving "OR" Statements

$ZKPoK\{w : R(x_0, w) = 1 \text{ or } R(x_1, w) = 1\}$



w is a valid R-witness only for x_b but V should not know which $b \in \{0,1\}$

pick $e_{1-b} \leftarrow \$\{0,1\}^t$

run Sim on (x_{1-b}, e_{1-b})

simulated transcript

$(a_{1-b}, e_{1-b}, z_{1-b})$

$a_b \leftarrow A_{x_b}(w, r)$

$\xrightarrow{(a_0, a_1)}$

$\xleftarrow{e \leftarrow \$\{0,1\}^t}$

$e_b = e \oplus e_{1-b}$

$z_b \leftarrow Z_{x_b}(w, r, e_b)$

$\xrightarrow{\{(e_0, e_1), (z_0, z_1)\}}$

check all:

$V_{x_0}(a_0, e_0, z_0) = 1\,?$

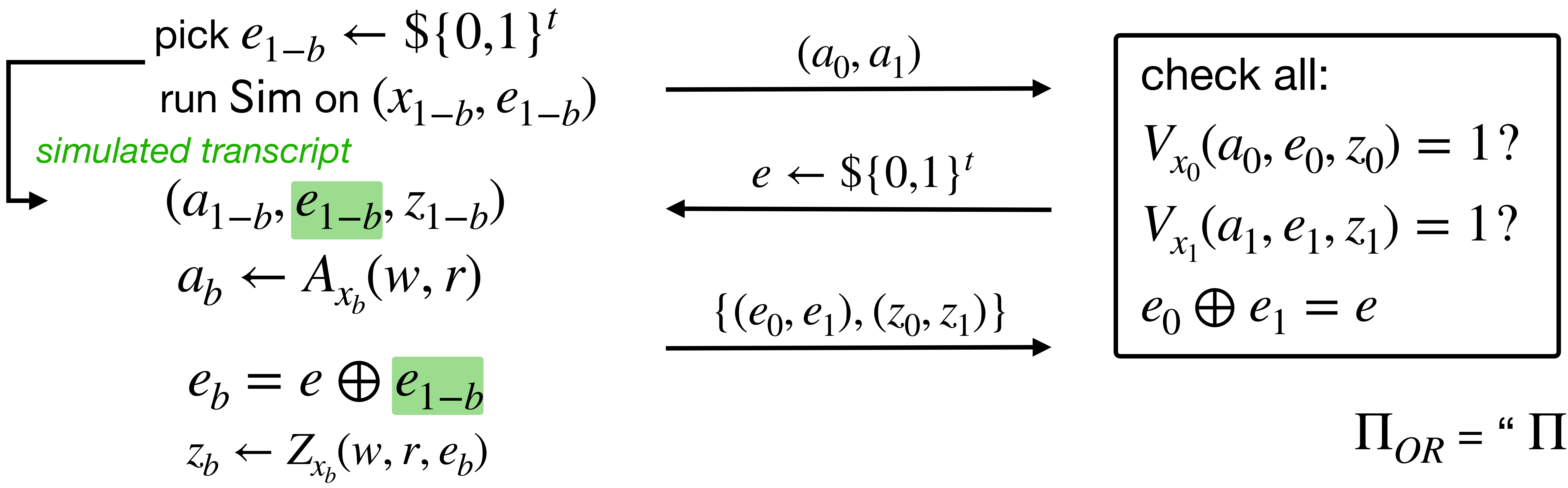
$V_{x_1}(a_1, e_1, z_1) = 1\,?$

$e_0 \oplus e_1 = e$

🧐 *The Trick* : P completes the protocol for the instance x_b that is true and "fakes" a proof for the other statement by running the simulator (in a clever way)

Proving “OR” Statements

$ZKPoK\{w : R(x_0, w) = 1 \text{ or } R(x_1, w) = 1\}$



pick $e_{1-b} \leftarrow \$\{0,1\}^t$

run Sim on (x_{1-b}, e_{1-b})

simulated transcript

$(a_{1-b}, e_{1-b}, z_{1-b})$

$a_b \leftarrow A_{x_b}(w, r)$

$e_b = e \oplus e_{1-b}$

$z_b \leftarrow Z_{x_b}(w, r, e_b)$

$\xrightarrow{(a_0, a_1)}$

$\xleftarrow{e \leftarrow \$\{0,1\}^t}$

$\xrightarrow{\{(e_0, e_1), (z_0, z_1)\}}$

check all:

$V_{x_0}(a_0, e_0, z_0) = 1\,?$

$V_{x_1}(a_1, e_1, z_1) = 1\,?$

$e_0 \oplus e_1 = e$

Π_{OR} = “ $\Pi_0 \vee \ \Pi_1$”

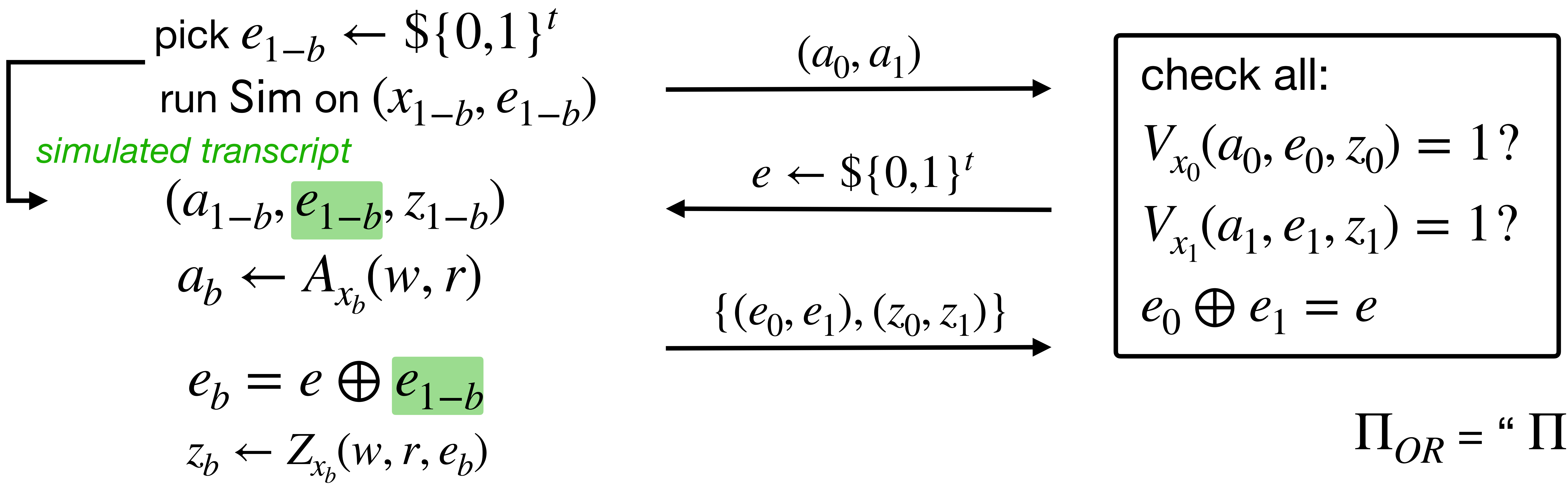
Completeness

follows from the completeness of $\Pi_0 = (A_{x_0}, Z_{x_0}, V_{x_0})$ & $\Pi_1 = (A_{x_1}, Z_{x_1}, V_{x_1})$

Special soundness: There exists a PPT algorithm $\mathscr{E}$ (extractor) that given **any** x and any **pair of accepting transcripts** $(a, e, z), (a, e', z')$ for x, **with** $e \neq e'$, outputs w such that $(x, w) \in R$.

Proving "OR" Statements

$ZKPoK\{w : R(x_0, w) = 1 \text{ or } R(x_1, w) = 1\}$



pick $e_{1-b} \leftarrow \$\{0,1\}^t$

run Sim on (x_{1-b}, e_{1-b})

simulated transcript

$(a_{1-b}, e_{1-b}, z_{1-b})$

$a_b \leftarrow A_{x_b}(w, r)$

$e_b = e \oplus e_{1-b}$

$z_b \leftarrow Z_{x_b}(w, r, e_b)$

$\xrightarrow{(a_0, a_1)}$

$\xleftarrow{e \leftarrow \$\{0,1\}^t}$

$\xrightarrow{\{(e_0, e_1), (z_0, z_1)\}}$

check all:

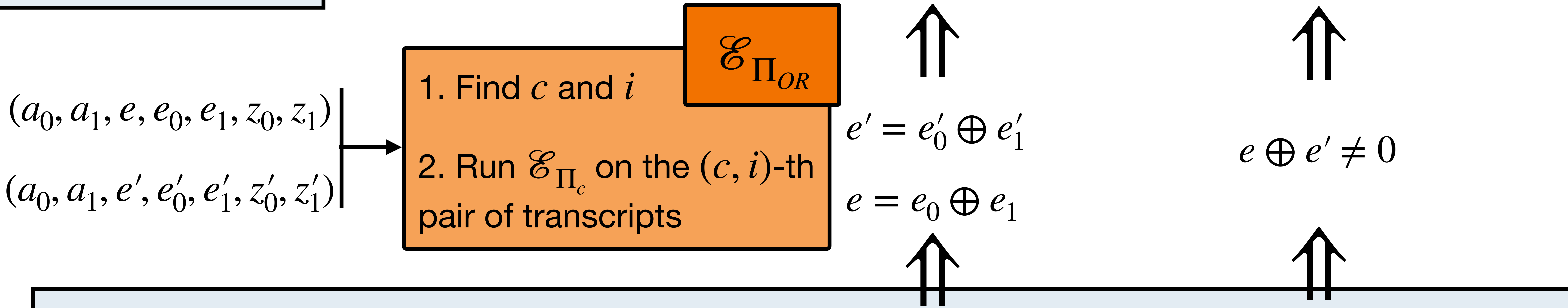
$V_{x_0}(a_0, e_0, z_0) = 1\,?$

$V_{x_1}(a_1, e_1, z_1) = 1\,?$

$e_0 \oplus e_1 = e$

Π_{OR} = " $\Pi_0 \vee \Pi_1$"

Special Soundness



$\exists c \in \{0,1\} \wedge \exists i \in \{1,\ldots,t\}$ s.t. $e_c[i] \neq e'_c[i]$

$(a_0, a_1, e, e_0, e_1, z_0, z_1)$

$(a_0, a_1, e', e'_0, e'_1, z'_0, z'_1)$

$\mathscr{E}_{\Pi_{OR}}$

1. Find c and i
2. Run $\mathscr{E}_{\Pi_c}$ on the (c, i)-th pair of transcripts

$\Uparrow$

$e' = e'_0 \oplus e'_1$

$e = e_0 \oplus e_1$

$\Uparrow$

$\Uparrow$

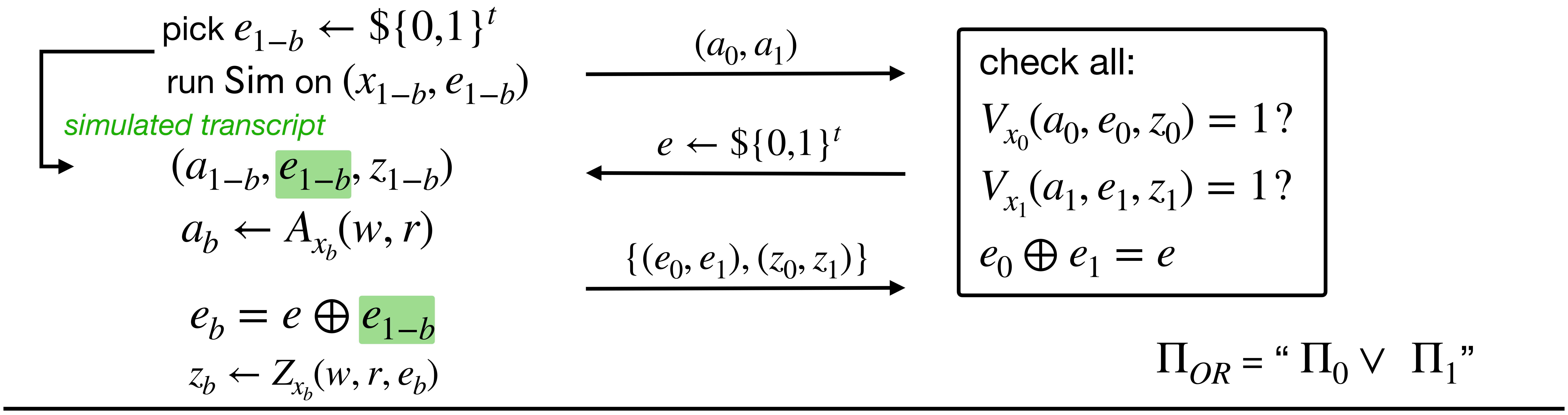
$e \oplus e' \neq 0$

$\Uparrow$

Special soundness: [..] given any **pair of accepting transcripts with** $e \neq e'$, **outputs** w

Proving “OR” Statements

$ZKPoK\{w : R(x_0, w) = 1 \text{ or } R(x_1, w) = 1\}$



pick $e_{1-b} \leftarrow \$\{0,1\}^t$

run Sim on (x_{1-b}, e_{1-b})

simulated transcript

$(a_{1-b}, e_{1-b}, z_{1-b})$

$a_b \leftarrow A_{x_b}(w, r)$

$e_b = e \oplus e_{1-b}$

$z_b \leftarrow Z_{x_b}(w, r, e_b)$

(a_0, a_1)

$e \leftarrow \$\{0,1\}^t$

$\{(e_0, e_1), (z_0, z_1)\}$

check all:

$V_{x_0}(a_0, e_0, z_0) = 1\,?$

$V_{x_1}(a_1, e_1, z_1) = 1\,?$

$e_0 \oplus e_1 = e$

$\Pi_{OR} = \text{“}\,\Pi_0 \vee \Pi_1\text{”}$

Special HV ZK **(Not needed for the exam)**

follows from using Sim_0 and Sim_1on inputs e_0 and e_1 respectively, where these are chosen at random subject to $e_0 \oplus e_1 = e$ for the string e received in input by Sim_{OR}.

Finally, the probability distribution over transcripts is independent of the branch b P is using for the real ZK Proof